VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÝ ANOTÁTOR DOKUMENTŮ

WEB DOCUMENT ANNOTATOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ HRÚZ**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. RADEK BURGET, Ph.D.**
SUPERVISOR

BRNO 2022

Ústav informačních systémů (UIFS) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Hrúz Tomáš**
Program: Informační technologie
Název: **Webový anotátor dokumentů**
Web Document Annotator
Kategorie: Web

Zadání:

1. Prostudujte současné technologie pro tvorbu klientských webových aplikací v jazyce JavaScript a existující řešení pro anotaci webových dokumentů.
2. Seznamte se s experimentálním nástrojem FitLayout, jeho aplikačním rozhraním a způsobem reprezentace dokumentů.
3. Navrhněte klientskou aplikaci pro zobrazení dokumentů, vyznačení jejich významných částí a jejich anotaci uživatelem.
4. Po dohodě s vedoucím implementujte navržené řešení pomocí vhodně zvolených technologií.
5. Proveďte testování vytvořené aplikace na množině testovacích dokumentů.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Lasila, I., Swick, R. R.: Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation 22 February 1999, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222
- The W3C SPARQL Working Group: SPARQL 1.1 Overview, W3C Recommendation 21 March 2013, https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015
- Rámec FitLayouthttps://github.com/FitLayout

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Burget Radek, doc. Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 11. října 2021

Abstrakt

Cieľom tejto práce je porovnať nástroje na anotáciu webových dokumentov a následne vytvoriť modul v experimentálnom nástroji FitLayout, ktorý bude implementovať požadovanú funkcionalitu pre anotácie. Riešenie bolo vypracované pomocou progresívneho frameworku Vue.js a vedomostí získaných pri porovnaní existujúcich riešení. Výstupom práce je nový samostatný komponent a ďalší upravený komponent v aplikácii FitLayout. Výsledkom je modul webovej aplikácie pre vytváranie nových oblastí v dokumentoch a upravený modul určený na správu komentárov a značiek dokumentu.

Abstract

The goal of this bachelor thesis is to compare of tools for annotation of web documents and create module for experimental tool FitLayout, which implements required functionality for annotation. Solution was developed with the progressive framework Vue.js and with knowledge obtained by comparison of existing annotation tools. The output of the work is new stand-alone component and other component with added functionality in FitLayout application. The result of the work is the web application module for creating new areas in documents and adjusted module for managing comments and tags.

Kľúčové slová

Webová aplikácia, Anotátor dokumentov, Segmentácia textu, Zoskupenie oblastí, Strom oblastí, JavaScript, Vue.js, PrimeVue, Frontend, FitLayout, GitHub, Java, Framework

Keywords

Web application, Document annotation, Text segmentation, Area grouping, Area tree, JavaScript, Vue.js, PrimeVue, Frontend, FitLayout, Github, Java, Framework

Citácia

HRÚZ, Tomáš. *Webový anotátor dokumentů*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Radek Burget, Ph.D.

Webový anotátor dokumentů

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením doc. Ing. Radka Burgeta PhD. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Tomáš Hrúz
27. apríla 2022

Poďakovanie

Rád by som poďakoval vedúcemu práce doc. Ing. Radkovi Burgetovi PhD. za odbornú pomoc a konzultácie, ktoré mi pomohli pri vypracovaní tejto práce.

Obsah

Kapitola 1

Úvod

Táto bakalárska práca rieši problematiku anotácie webových dokumentov a tvorbu webových aplikácii. Výsledkom by mali byť moduly v experimentálnej webovej aplikácii FitLayout, ktoré budú schopné vytvárať skupiny prvkov z webového dokumentu a vytvárať z nich jednotné celky. Tieto celky ale aj samostatné prvky bude následne možné komentovať a pridávať im kategórie, čo slúži na efektívnejšiu prácu s informáciami.

Téma anotácie je v dnešnom svete veľmi aktuálna, a to pretože žijeme v informačnej dobe. Je potrebné mať veľké množstvo informácii ale bez správnej organizácie sa stávajú bezcenné. Anotáciou textu sa vytvára obsah triedený do logických kategórii, s vyznačenými dôležitými časťami a prípadne s poznámkami za čiarou. Všetky tieto kroky vedú k zvyšovaniu efektivity učenia, čítania a celkového vstrebávania informácii.

Trh s anotačnými aplikáciami je v dnešnej dobe plný vytvorených riešení, to ale neznamená, že nie je priestor na nové nápady. Ľudia už teraz potrebujú nástroje na lepšiu správu informácii, a s rýchlym pribúdaním stále nových informácii potrebujú a v budúcnosti budú stále viac potrebovať spracovať a organizovať dokumenty.

Prácu som si vybral z dôvodu, že mám skúsenosti s tvorbou užívateľských rozhraní a vytvorenie webovej aplikácie mi umožňuje zoznámiť sa s používanými technológiami. Trh s webovými aplikáciami momentálne rastie a pravdepodobne sa v budúcnosti stane výrazne dominantným spôsobom implementácie nástrojov všetkého druhu. Preto si myslím, že táto práca bude veľkým prínosom v budúcom kariérnom živote a navyše môže obohatiť aj vedomosti o organizácii textu a efektívnejšom učení.

Text práce tvorí sedem kapitol. Prvá kapitola 1 je tento úvod kde je zhrnutá motivácia a uvedená téma tejto práce.

V druhej kapitole 2 sú vysvetlené základné technológie pre tvorbu webových aplikácii. Kapitola je rozdelená na dve základné sekcie frontend 2.1 a backend 2.2. Frontend obsahuje základné technológie pre tvorbu webov ale aj progresívne frameworky pre jazyk JavaScript. Backend obsahuje základné technológie použité pre serverovú časť webových aplikácii ale nepreberá ich viac do hĺbky, pretože táto práca rieši hlavne frontend.

V tretej kapitole 3 sú predvedené existujúce riešenia na anotáciu dokumentov. Každý nástroj je testovaný a popísaný z kladného aj záporného hľadiska.

Štvrtá kapitola 4 obsahuje návrh riešenia. V úvode obsahuje sekciu FitLayout 4.1, ktorá popisuje aplikáciu, do ktorej budú implementované požiadavky tejto práce. Ďalej sú zhrnuté požiadavky na funkcionalitu 4.2 a následný návrh riešenia 4.3.

V kapitole päť 5 je popis konkrétnej implementácie. V prvej sekcii 5.1 je popísaný výber technológii a v druhej sekcii 5.2 sú popísané konkrétne komponenty, v ktorých bol implementovaný kód. Najzaujímavejšia je podsekcia Selection, ktorá obsahuje implementáciu

nového komponentu pre tvorenie nových oblastí a podsekcia AnnotationPanel s pridanou funkcionalitou pre anotácie.

V šiestej kapitole 6 je popísané testovanie počas vývoja ale aj testovanie hotovej aplikácie. Táto kapitola obsahuje aj sekciu kde je odkaz na online verziu nástroja 6.1.

Ako posledná je kapitola sedem 7, ktorá zahrňuje zhodnotenie výsledku, návrhy na ďalší vývoj aplikácie a záverečné slová.

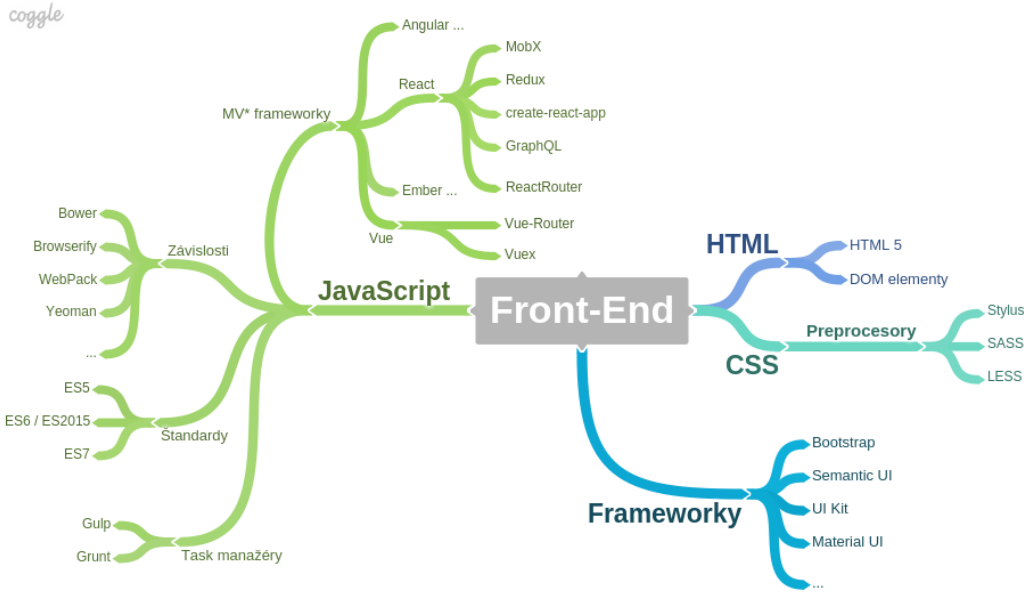
Kapitola 2

Súčasné technológie pre tvorbu webových aplikácii

Táto kapitola popisuje súčasné technológie na tvorbu webových aplikácii. Delí sa na dve hlavné sekcie Frontend a Backend. V prvej sekcii sa rozoberajú základné jazyky a prístupy a potom aj konkrétne nadstavby a ich porovnanie. V Druhej sekcii sú okrajovo popísané nástroje ale neprechádzajú sa do hĺbky, pretože pre úlohou tejto práce je hlavne práca s frontendom a informácie o backende sú doplnkové pre pochopenie súvislostí.

2.1 Frontend

Frontend je časť aplikácie, ktorá sa stará o vzhľad grafického rozhrania a interakciu s užívateľom. Môže sa označovať aj pojmom klientska časť aplikácie. Medzi dôležité aspekty vývoja frontendu patrí: dôraz na použiteľnosť, pretože je dôležité aby operácie vykonávané s rozhraním dávali zmysel aj bez rozsiahlych návodov na používanie. V dnešnej dobe je veľmi dôležité vyriešiť aj problém prístupnosti pre zdravotne hendikepovaných. To znamená napríklad výber vhodnej farebnej palety pre farboslepých ľudí alebo pridávanie popisov k obrázkom a tlačidlám pre možnosť hlasného čítania obsahu[3]. A v neposlednej rade je podstatný aj vzhľad, pretože pekné veci predávajú a pri aplikáciách ide o to aby boli úspešné a prinášali zisk. Spojením týchto a ešte ďalších aspektov vzniká kvalitné užívateľské rozhranie, ktoré ma šancu sa presadiť na dnešnom trhu.



Obr. 2.1: Pojmová mapa termínov spojených s frontendom. Prevziate od Lubomir Herko [6]

2.1.1 HTML

HTML[13] je základný stavebný blok webových stránok a aplikácii. Hlavnou úlohou je určovanie štruktúry a významu zobrazovaného obsahu. Skratka HTML znamená *HyperText Markup Language*, z čoho vyplývajú dva podstatné fakty. Jedná sa o značkovací jazyk, čo znamená, že používa značky(*tagy*) na popis štruktúry. Značka sa zapisuje v tvare `<typ značky a voliteľné atribúty>`. Typ značky nerozlišuje veľké a malé písmená a atribúty sa oddeľujú medzerou. Tieto značky sú písané s hlavným textom a obaľujú ho medzi začiatočnou a ukončovacou značkou[8]. Hypertext je text, ktorý obsahuje odkazy na iné texty. Pojem hypertext vytvoril Ted Nelson, pôvodom Americký sociológ a filozof, ktorý sa pohyboval vo svete informačných technológii. V dnešnej dobe a v kontexte internetu hypertext označuje texty, ktoré navzájom prepájajú dokumenty z celého internetu a nie len v rámci jednej webovej stránky. Všeobecnejší pojem, ktorý zahŕňa text, grafiku, video a zvuk je pojem hypermedia[22].

2.1.2 CSS

CSS[12] je štýlový jazyk, ktorý slúži na popis toho ako budú prvky v HTML, XML alebo iných značkovacích jazykoch vyzerať. To zahŕňa veci ako veľkosť, farba, viditeľnosť alebo umiestnenie na webovej stránke. Skratka CSS znamená *Cascading Style Sheets* alebo kaskádové štýly. Toto pomenovanie vyplýva z charakteristiky jazyka. Využíva kaskádový algoritmus na určenie toho, ktorý štýl sa vyberie pre daný prvok. Štýly môžu pochádzať z viacerých zdrojov a algoritmus ich prechádza ako kaskády a vyberie v hierarchii ten posledný a nastaví ho. Pôvody delíme na: Štýl agenta používateľa(*User-agent stylesheet*), štýl autora(*Author stylesheet*) a štýl užívateľa(*User stylesheet*). Štýl agenta používateľa je implementovaný priamo v internetovom prehliadači. Štýl autora je najbežnejší a jedná sa o štýl, ktorý definuje autor webovej stránky. Štýl užívateľa môže byť prepísaný priamo užívateľom

aby vyhovoval jeho potrebám. Zoradenie pravidiel podľa dôležitosti je vzostupne v poradí: agent, užívateľ, autor, kde posledný spomenutý ma najväčšiu prioritu. Ďalší faktor, podľa ktorého vyberáme prioritu je umiestnenie v kóde. Vzostupne od štýlu implementovaného v externom súbore, následne implementácia priamo v súbore s hlavným obsahom a nakoniec štýl priamo na riadku. Znamená to, že keď sa nastaví farba textu na červenú riadkovým štýlom a na čiernu externým štýlom, tak finálny text bude zobrazený ako červený. Formát zápisu pravidla potrebuje selektor, ktorý určí pre aký prvok sa štýl nastavuje, a akú vlastnosť a jej hodnotu dostane.

2.1.3 JavaScript

JavaScript[5] je interpretovaný programovací jazyk so základmi objektovo orientovaného programovania. Je súčasťou webových prehliadačov. Jednou z jeho hlavných úloh je práca na klientskej strane webovej stránky ale sú aj prípady kedy sa JavaScript používa aj na webových serveroch. Umožňuje tvorbu dynamických webov, kde obsah nie je len statický HTML dokument ale interaktívne miesto, ktoré má animácie, dokáže zobrazovať dynamický obsah, ako napríklad hodiny alebo umožňuje užívateľovi komunikáciu so stránkou. JavaScript pripomína programovými konštrukciami iné programovacie jazyky ako C, C++ alebo Java. V iných veciach ale podobnosť nie je. JavaScript nemá typovú kontrolu, čo znamená že premenná nemusí mať určený konkrétny typ. Mnohé myšlienky preberá z jazyka Perl, odkiaľ využíva napríklad prácu s regulárnymi výrazmi alebo prácu s poľami.

Klientsky JavaScript

Jedná sa o prekladač jazyka JavaScript vložený do Webového prehliadača, ktorý zároveň obsahuje aj jadro jazyka JavaScript. Medzi užívateľmi je toto ta najbežnejšia verzia s ktorou sa stretnú. Kombinuje možnosť spracovania skriptov interpretom JavaScriptu s objektovým modelom dokumentov(DOM), ktorý je definovaný webovým prehliadačom. Klientsky JavaScript sa považuje za "*srdce*" dynamických HTML dokumentov.

Programovanie

Pre vloženie JavaScript kódu do HTML dokumentu je potrebné použiť značky `<script></script>`, v ktorých je napísaný kód. JavaScript je možné volať aj z externého súboru ktorý bude obsahovať kód.

```
<!DOCTYPE html>
<html>
    <body>
        <h2>JavaScript in HTML</h2>

        <p id="demo">Hello World.</p>

        <button type="button"
        onclick='document.getElementById("demo").innerHTML = "Hello JavaScript"'>
        Click me
        </button>
    </body>
</html>
```

Výpis 2.1: Príklad použitia JavaScript kódu priamo v zdrojovom texte HTML dokumentu

```
<!DOCTYPE html>
<html>
    <head>
        <script type="text/javascript" src="message.js"></script>
    </head>
    <body>
        <h2>Javascript from external file</h2>
        <form>
            <input type="button" value="click" onclick="msg()"/>
        </form>
    </body>
</html>
```

```
function msg(){
    alert("Hello JavaScript")
}
```

Výpis 2.2: Príklad použitia JavaScript kódu z externého súboru s príponou `.js`. Horná časť kódu je v HTML súbore a odsadená časť je v JavaScript súbore

Spoločne s použitím objektového modelu, ktorý umožňuje pristupovať k jednotlivým prvkom HTML dokumentu sa z JavaScriptu stáva mocný nástroj, ktorý môže vygenerovať aj celý HTML dokument ak je to potrebné, prípadne zmeniť ľubovoľný obsah na stránke. Dokáže pracovať s HTML formulármi a čítať alebo zapisovať ich hodnoty. Využitie môže byť napríklad v prípade, že je potrebné kontrolovať vstupy od užívateľa a namiesto toho aby sa všetko posielalo na server, tak JavaScript skontroluje či sú vstupy správne a umožní ich poslanie až po správnych vstupoch. Tým sa zamedzí zbytočnej komunikácii so serverom, ktorý by chybné požiadavky aj tak nemohol spracovať. Ďalšia dôležitá vlastnosť JavaScriptu je definícia ovládačov udalostí. Znamená to, že pokiaľ nastane nejaká udalosť vykoná sa nadefinovaný kód. Takáto udalosť môže byť napríklad kliknutie tlačidla, alebo prechod myši ponad hypertextový odkaz. Jedná sa o veľmi podstatnú vlastnosť, pretože grafické

aplikácie sú založené na interakcii s užívateľom a ich programovanie vyžaduje model riadený udalosťami.

Nedostaky

- Nedokáže generovať grafický obsah, okrem možnosti generovať HTML obsah
- Neumožňuje prácu so súbormi, ich čítanie a zapisovanie, čo je v istom ohľade dobrá vec, pretože sa nestane, že by sa pri prístupe na nebezpečnú webovú stránku útočník dostal k našim súborom
- Nemá nástroje na sieťovú komunikáciu

Jadro JavaScript

- Pre zápis využíva 16 bitovú znakovú sadu UNICODE, ktorý používa prakticky všetky znaky na planéte
- Rozlišuje veľké a malé písmená, čiže pokiaľ je premenná alebo kľúčové slovo malými písmenami, tak s veľkými písmenami sa jedna o iný prvok
- Príkazy sú zakončené bodkočiarkou, avšak JavaScript umožňuje príkazy bez takéhoto zakončenia. Nie je to ale dobrá praktika
- Základné primitívne dátové typy: čísla, reťazce, logické hodnoty
- Zložený dátový typ: objekt
- Objekt je nezoradená kolekcia pomenovaných hodnôt alebo zoradená kolekcia očíslovaných hodnôt čo sa nazýva aj pole
- Funkcia: objekt, ktorý má priradený vykonateľný kód
- Netypový jazyk: premennej je možné priradiť ľubovoľný typ a súvisí to aj s automatickým prevádzaním hodnôt z jedného typu na druhý. Napríklad prevod reťazca na číslo
- Využíva automatické uvoľňovanie pamäti(garbage collection)

Node.js

Node.js[2] je prostredie na spúšťanie JavaScript kódu a umožňuje spúšťanie programov priamo na počítači a nie v prehliadači. Je založený na V8, čo je JavaScript engine, používaný prehliadačom Google Chrome.

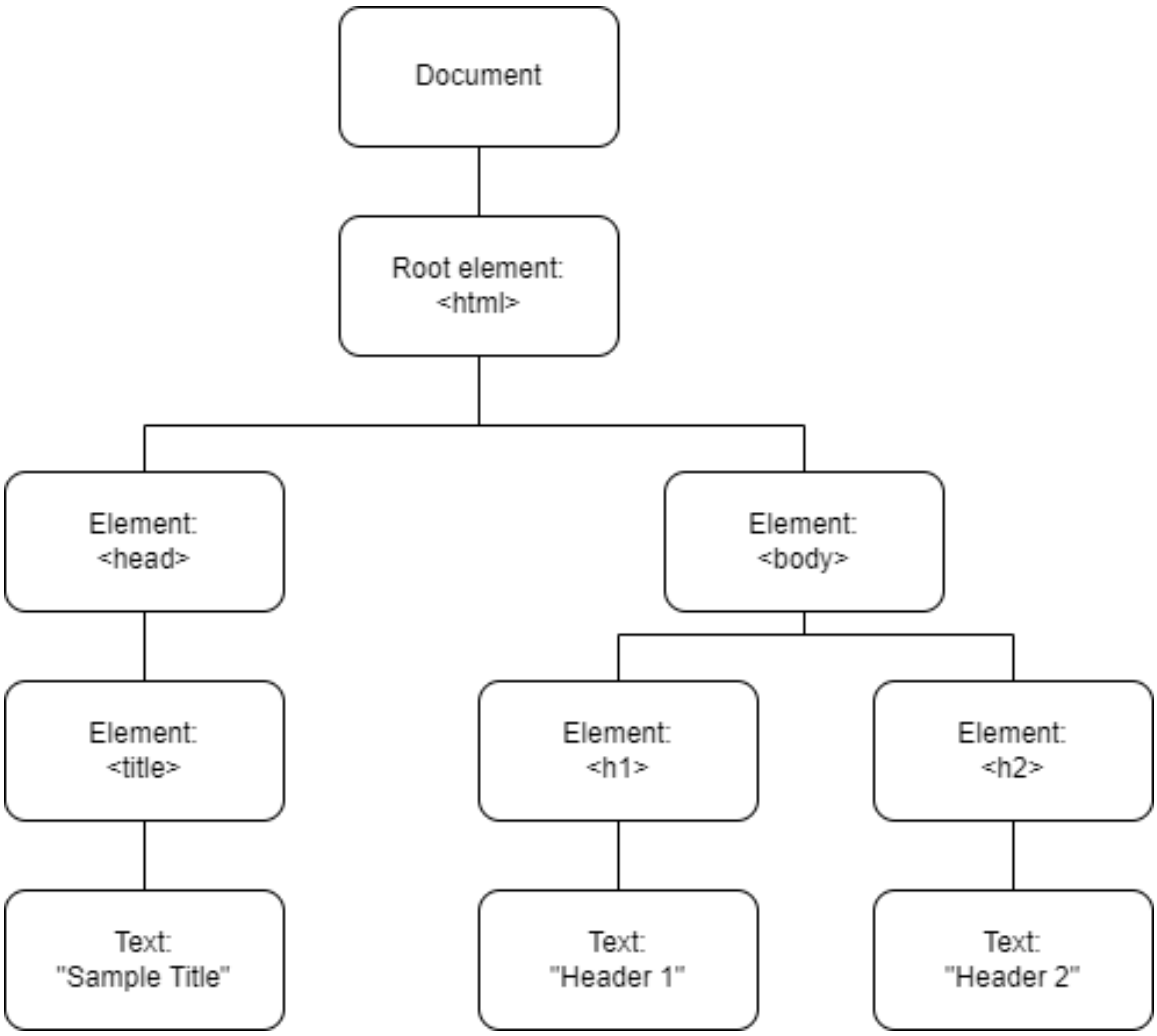
npm

Je to nástroj na správu balíčkov pre platformu Node.js. Stará sa o ukladanie balíčkov a ich následné vyhľadávanie. Skladá sa z troch častí. Webová stránka, kde sa dajú objavovať a spravovať balíčky. Príkazový riadok slúži na interakciu s užívateľmi ako je sťahovanie a inštalácia balíčkov. Posledná súčasť je register, čo je verejná databáza JavaScript softvéru a informácii k nemu[1].

[1] https://docs.npmjs.com/about-npm

2.1.4 DOM

Alebo Objektový model dokumentu je aplikačné programovacie rozhranie pre reprezentáciu dokumentu a prístup k jeho prvkom a manipulácii s týmito prvkami. V prípade dokumentu HTML to môže byť manipulácia so značkami. DOM reprezentuje dokument ako stromovú štruktúru, kde uzly stromu odpovedajú typom obsahu v dokumente ako napríklad spomínané značky a reťazce textu[5].



Obr. 2.2: Ukážka HTML dokumentu v reprezentácii DOM

Terminológia uzlov je v celku jednoduchá. Uzol priamo nad daným uzlom je rodičovský uzol. Uzol priamo pod daným uzlom je potomok daného uzlu a uzli na rovnakej úrovni sú súrodenci. Množina všetkých uzlov nad tvorí dokopy predkov a naopak množina uzlov pod prvkom sú nasledovníci. Na vrchole celého stromu sa nachádza koreňový uzol. K prístupu k týmto uzlom slúži rozhranie Node, ktoré definuje vlastnosti a metódy manipulácie so stromom.

Vlastnosti:

- **childNodes**

- **firstChild**
- **lastChild**
- **nextSibling**
- **previousSibling**
- **parentNode**

Metódy:

- `appendChild()`
- `removeChild()`
- `replaceChild()`
- `insertBefore()`

Uzly majú vlastnosť nodeType, ktorá špecifikuje o aký typ uzlu sa jedná.

- **Element**
- **Text**
- **Document**
- **Comment**
- **DocumentFragment**
- **Attr**

Rozhranie môže pristupovať aj k atribútom prvkov a to metódami.

- `getAttribute()`
- `setAttribute()`
- `removeAttribute()`

2.1.5 Angular

Angular[9] je aplikačný rámec s otvoreným kódom na tvorbu webových aplikácii a dynamických webov. Je vytvorený spoločnosťou Google, čo vytvára pre tento rámec veľké zázemie. Jedná sa o následovníka AngularJS, ktorý bol založený na inom princípe ako Angular. Moderný Angular je postavený na princípe znovu použiteľných komponent.

2.1.6 React

React[20] je aplikačný rámec s otvoreným kódom na tvorbu webových aplikácii a dynamických webov. Vytvorila ho spoločnosť Facebook a tak ako Angular je založený na princípe znovu použiteľných komponent a prišiel s tým skôr ako Angular.

2.1.7 Vue.js

Vue.js[10] je aplikačný rámec (*framework*) pre JavaScript, vytvorený v roku 2014 vývojárom Evanom You. Počas vytvorenia tohto projektu pracoval vo firme Google, kde používal aplikačný rámec AngularJS, z ktorého Vue.js vychádza. Myšlienkou bolo zobrať z Angularu veci, ktoré sa mu páčili a vytvoriť odľahčený rámec pre rýchle a ľahké vytváranie bohatých webových rozhraní. Z Vue.js sa rýchlo stal jeden z najpoužívanejších aplikačných rámcov pre tvorbu webových aplikácii, čo potvrdzuje aj fakt, že je na portáli GitHub ako jeden z najobľúbenejších repozitárov. Existuje ako otvorený softvér, takže je voľne dostupný na stiahnutie a používanie. Aktuálna verzia Vue.js je 3.

Komponenty

Je to samostatný kód, ktorý reprezentuje časť stránky. Komponent obsahuje svoje vlastné dáta, JavaScript a vlastný štýl. To je obrovská výhoda ako z hľadiska prehľadnosti tak aj z hľadiska funkcionality. Každá časť stránky je oddelená a má svoj vlastný kód, čo sprehľadňuje kód a zároveň nám zaručuje, že nebude negatívne ovplyvňovať kód v iných komponentoch. Keď je to však potrebné sú schopné komunikovať so zvyškom kódu ale riadenou cestou.

```
<template>
    <!-- HTML CODE -->
</template>

<script>
    // JAVASCRIPT WITH OPTIONS OR COMPOSITION API
</script>

<style>
    /* CSS CODE */
</style>
```

Výpis 2.3: Ukážka kostry komponentu Vue

Toto je zjednodušená kostra Vue.js komponenty. Ako prvé obsahuje časť `<template>`, v ktorej sa nachádza klasický HTML obsah obsahujúci značky. Sekcia `<style>` je obsahovo rovnaká ako klasické CSS so selektormi a nastavenými štýlmi. Časť `<script>` s JavaScriptom je však dosť špecifická pre Vue.js.

```
// Options API
export default {
  data() {
    return {
      name: 'John',
    };
  },
  methods: {
    doIt() {
      console.log(`Hello ${this.name}`);
    },
  },
  mounted() {
    this.doIt();
  },
};
```

Výpis 2.4: Ukážka Options API script sekcie [15]

Tento spôsob zápisu sa nazýva Options API a bol používaný ešte v druhej verzii Vue. S verziou 3 však prišiel nový zápis s názvom Composition API.

```
// Composition API
export default {
  setup() {
    const name = ref('John');

    const doIt = () => console.log(`Hello ${name.value}`);

    onMounted(() => {
      doIt();
    });

    return { name };
  },
};
```

Výpis 2.5: Ukážka Composition API script sekcie [15]

Vo Vue 3 je stále možné používať Options API ale odporúča sa migrovať na nový spôsob zápisu, ktorý zlepšuje jeho použitie.

PrimeVue

PrimeVue[2] je knižnica prvkov grafického rozhrania vytvorená spoločnosťou PrimeTek Informatics ako otvorený kód. Poskytuje viac než 80 komponent, návrh vlastných grafických tém alebo použitie prednastavených tém. Inštalácia a pridanie do projektu je jednoduché.

```
⇒ npm install primevue@^3 -save

⇒ npm install primeicons -save
```

[2] https://www.primefaces.org/primevue/

Do projektu sa následne pridá príkazom:

```
⇒ import PrimeVue from 'primevue/config';
```

2.1.8 Porovnanie Angular, React a Vue.js

Z hľadiska popularity dlhú dobu vyhrával Angular a to vďaka tomu, že vznikol podstatne skôr a na trhu nebola žiadna väčšia konkurencia. React si po svojom vzniku začal získavať veľké publikum, až kým neprekonal Angular. S príchodom aplikačného rámca Vue sa tento rast spomalil a Vue vytvorený jednotlivcom začal konkurovať gigantom ako Google a Facebook. Neponúkal tak veľké množstvo možností ako jeho starší konkurenti ale kombinoval ich dobré vlastnosti a to znamenalo nárast popularity.

Autor Yuriy Vovchuk tieto jazyky porovnal bližšie vo svojej bakalárskej práci *"Srovnání progresivních frameworků Vue.js, React a Angular"*[21]. V jeho porovnaní sa nachádza aj technológia jQuery, ktorá v tejto súvislosti nie je dôležitá a preto ďalej nebude spomínaná. Hodnotil ho v niekoľkých kategóriách so vzorkou vývojárov. Pred hodnotením ešte zisťoval ich oboznámenie s technológiami, kde najväčšie skúsenosti mali s Angularom, nasledoval React a až potom Vue.js.

Čitateľnosť kódu

V kategórii čitateľnosti kódu sa hodnotila zrozumiteľnosť zdrojového kódu a jeho pochopenie. s prehľadom vyhral Vue.js, zatiaľ čo React a Angular dosiahli takmer identické hodnotenie od opýtaných vývojárov. Na Vue.js chválili výbornú štruktúru a ľahké pochopenie a to aj napriek tomu, že niektorí o Vue.js počuli prvý krát.

Efektivita kódu

Efektivita kódu rieši množstvo použitých znakov, na dosiahnutie rovnakého výsledku. Čím menší počet znakov, tým lepšia efektivita. Počet znakov bol rozdelený na časti HTML a JavaScript. React mal veľký náskok v počte znakov HTML kódu ale v celkovom počte zaostával viac než dvojnásobne oproti Vue.js a Angular skončil tesne za Vue.js.

Rýchlosť

V tejto kategórii je dôležitá rýchlosť vykonávania kódu. Opäť viedol Vue.js a to s viac než dvojnásobnou rýchlosťou oproti konkurentom. React a Angular mali takmer identické výsledky, pričom Angular bol rýchlejší o pár percent.

Dokumentácia

Hodnotí do akej miery bol vysvetlený text zrozumiteľný. Vue.js získal prvenstvo, za ním nasledoval React a na konci bol s väčším rozdielom Angular.

Ekosystém

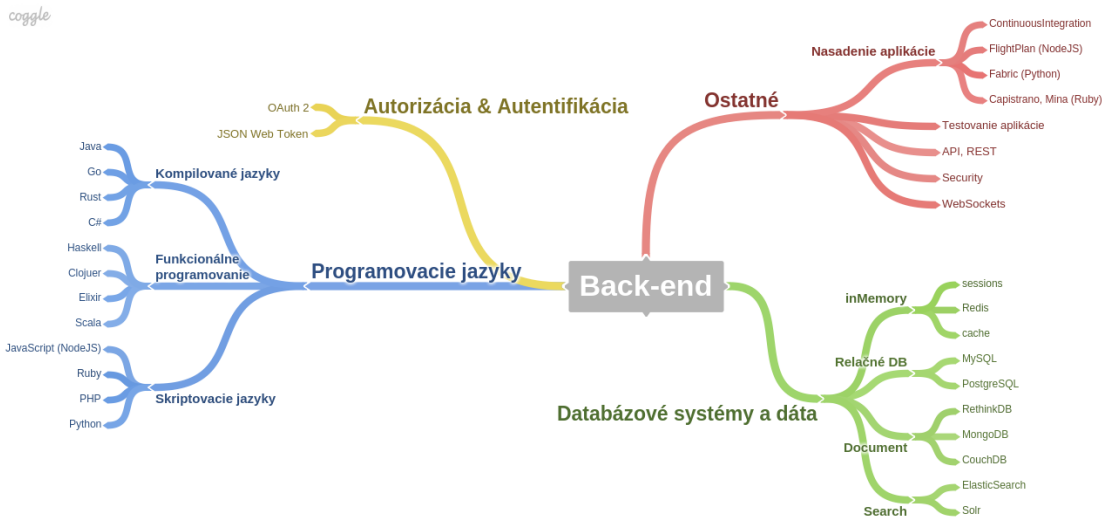
Ekosystém zahŕňa rozsiahlosť používania medzi programátormi, čo vedie napríklad k rýchlejšiemu hľadaniu odpovedí na problémy. V tejto kategórii ako v jedinej neviedol Vue.js ale React. Vue.js však nasledoval tesne za Reactom a s väčším rozdielom za nimi bol Angular.

Záver

Celé hodnotenie vyhral s veľkým náskokom Vue.js. React a Angular skončili s podobným hodnotením, pričom React vyšiel ako mierne lepšia možnosť.

2.2 Backend

Backend[11] je časť aplikácie, ktorá sa stará o veci, čo nie sú viditeľné pre užívateľa. Jeho úlohou je komunikácia so serverom a databázou, preto sa označuje aj ako serverová časť aplikácie. Užívateľ s backendom priamo nekomunikuje ale namiesto toho interaguje s frontendom a ten volá funkcie backendu. Hlavné úlohy backendu sú zamerané na prácu s dátami[17]. Prvou z týchto úloh je prístup k dátam. Užívateľ k nim musí byť schopný pristupovať keď odošle požiadavku, a musí k nim pristupovať bezpečne, hlavne v prípade finančných, zdravotných a osobných informácii. Ďalšou úlohou je transformácia a úprava dát. Napríklad v situáciách keď požiadavka potrebuje dáta z viacerých zdrojov a spojí ich do jednej odpovede. V neposlednej rade je potrebné tieto dáta užívateľovi vrátiť a frontend ich mohol zobraziť.



Obr. 2.3: Pojmová mapa termínov spojených s backendom. Prevziate od Lubomir Herko[6]

2.2.1 Java

Je to vysoko-úrovňový, objektovo orientovaný programovací jazyk[4]. Jedná sa o jeden z najpoužívanejších jazykov pre programovanie serverových častí aplikácii a veľké množstvo spoločností oceňovaných spoločnosťou Fortune 500 používa práve jazyk Java pre svoje webové servery. Základ jazyka tvorí aplikačné rozhranie pre programovanie, ktoré obsahuje viac než 3000 tried pre sieťovú komunikáciu, prácu s grafikou, databázami a mnoho ďalších. To nie je všetko a Java podporuje aj veľké množstvo bezplatných knižníc. Ďalšia dôležitá súčasť jazyka je JVM(*Java Virtual Machine*), čo je virtuálny stroj, vďaka ktorému nie je potrebné kód kompilovať pomocou rôznych prekladačov pre rôzne platformy. Preklad sa vykonáva iba raz a spustiteľný kód sa potom môže spustiť na ktoromkoľvek systéme za predpokladu, že je na ňom nainštalovaný JVM.

2.2.2 Python

Python je multiparadigmatický programovací jazyk[7], ktorý umožňuje písanie platformovo nezávislých programov, ktoré bežia na systéme Windows, Linux a OS X. Python má aj veľké zázemie vo svete vstavaných systémov. Je vhodný na rýchly vývoj malých skriptov ale aj na tvorbu robustných softwarových projektov. Má veľmi dobrú podporu objektovo orientovaného programovania ale zaostáva v rýchlosti napríklad za jazykom C, ktorý sa však pre vývoj webových aplikácii nepoužíva a rýchlosťou sa rovná iným jazykom použitým pre vývoj webových aplikácii. Výhodou jazyka je možnosť použiť knižnice písané v jazyku C ak je rýchlosť nevyhnutná ale zároveň, keď to nie je tak potrebné je možné použiť knižnice s veľkou úrovňou abstrakcie pre rýchle písanie kódu. Python je otvorený software, dostupný zdarma a to umožňuje upravovať kód pre vlastnú potrebu a to vedie k veľkému množstvu užívateľských knižníc.

2.2.3 PHP

PHP je multiplatformový, programovací jazyk[19], ktorý sa používa prevažne na tvorbu webov a na rozdiel od niektorých iných jazykov, ktoré sa dnes na tvorbu webov používajú, bolo PHP od začiatku vytvorené pre potreby tvorby webov. Je to open source projekt, ktorý prichádza už predinštalovaný na OS X a distribúciách operačného systému Linux, preto sa môže PHP považovať za dobrú možnosť pre začiatočníkov. Kód, ktorý beží na Linux serveri by mal fungovať bez zmeny aj na Windows serveri a je schopný fungovať s rôznymi HTTP servermi a tak isto s rôznymi databázami, ako napríklad MySQL, PostgreSQL, Oracle a ďalšie. Obvykle beží na nejakom webovom serveri, ku ktorému môžu pristupovať rôzny užívatelia pomocou internetového prehliadaču. Komunikácia prebieha spôsobom, že užívateľ zašle požiadavku na zobrazenie nejakého dokumentu "*example.php*"na webový server. Tam ju môže spracovať napríklad HTTP server Apache a ten si od enginu PHP vyžiada daný dokument. PHP engine prečíta súbor a vykoná príkazy v ňom napísané. Následne vráti svoj výstup a odošle ho späť na HTTP server, ktorý to prepošle späť užívateľovi a tomu sa obsah zobrazí vo webovom prehliadači. PHP kódy môžu obsahovať aj JavaScript, ten sa však vykoná až na strane klienta.

2.2.4 SPARQL

SPARQL[16] je dotazovací jazyk pre RDF[18], čo ho odlišuje od SQL ktoré sa dotazuje nad relačnými dátami a má výhodu, že dokáže komunikovať naraz s viacerými zdrojmi. RDF(*Resource Description Framework*)[18] je model pre výmenu dát na webe a je dôležitou súčasťou sémantického webu. Sémantický web by sa mal v budúcnosti stať nasledovníkom súčasného webu, kde budú informácie uložene štruktúrovane a štandardizovane, čo by viedlo k ľahšiemu vyhľadávaniu a spracovaniu dát na internete. SPARQL v základe slúži len na sťahovanie dát a nie nahrávanie a na prenos využíva protokol založený na HTTP.

Kapitola 3

Existujúce riešenia pre anotáciu dokumentov

V tejto kapitole sa čitateľ zoznámi s problematikou anotácie. Aké aspekty anotácie sú dôležité a na čo by sa vývojári mali zamerať pri tvorbe aplikácii na anotáciu dokumentov. V tejto kapitole sú analyzované a porovnané rôzne konkurenčné aplikácie určené na anotáciu.

3.1 Anotácia

Anotácia je úkon, ktorý pracuje s textom a informáciami. Môže slúžiť na zhrnutie obsahu v dokumente, môže to byť nástroj na zvýraznenie dôležitých slov alebo myšlienok alebo aj metóda lepšieho pochopenia textu a zlepšenie učenia. Všetky tieto body sú v dnešnom svete informácii veľmi dôležité. Anotácia môže prebiehať niekoľkými spôsobmi.

- Zhrnutie dôležitých pojmov pre alebo slov
- Písanie komentárov k sekciám obsahu
- Zaradenie obsahu do kategórii
- Grafické zvýrazňovanie

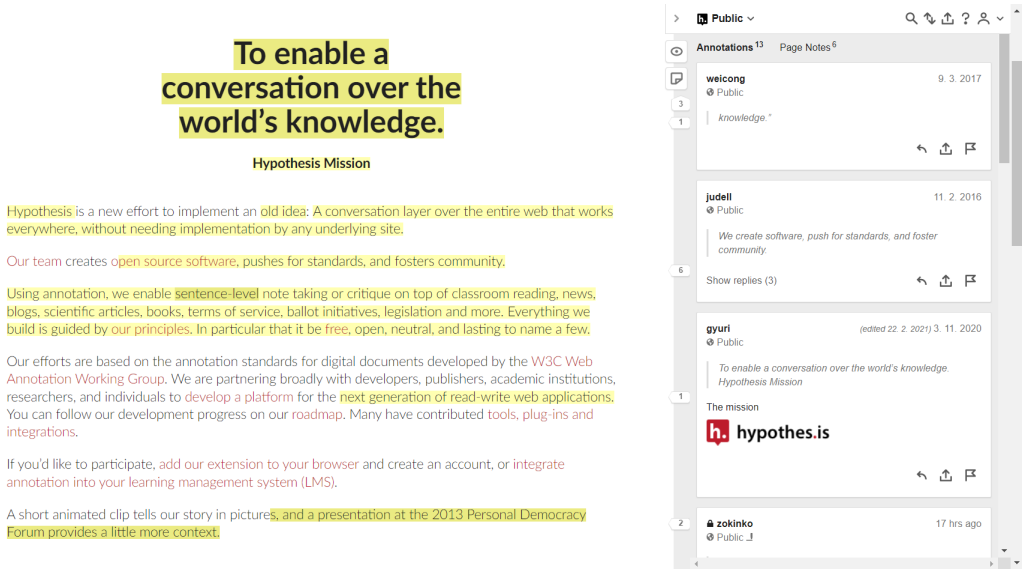
Tieto kroky vedú k interakcii s textom a človek je potom nútený rozmýšľať nad obsahom a môže ho lepšie pochopiť. Keď sa bude potom vracať k anotovanému textu po dlhšom čase tak si vybaví skôr dôležité body a myšlienky textu. Toto všetko má obrovské využitie v školstve aj na strane učiteľov, tak aj pri samoštúdiu žiakov. Veľké využitie je v pracovnom prostredí ale aj v súkromnom živote.

3.2 Hypothesis

Je to nástroj na anotáciu fungujúci pod licenciou *"FreeBSD license"*, a je využiteľný formou rozšírenia pre Google Chrome alebo aj ako API, ktoré môžu ľudia využívať pre potreby svojich webových aplikácii. Projekt[1] vznikol v roku 2011 pomocou finančnej podpory cez spoločnosť Kickstarter, ktorá sa zameriava na vyzbieranie cieľovej sumy od fanúšikov a všetkých ľudí, čo chcú podporiť začínajúce projekty. Hlavná myšlienka projektu Hypothesis je pojem „*Open annotation*“, čo je vrstva nad webovými stránkami spájajúca celý svet

[1] https://web.hypothes.is/about/

myšlienkami a komentármi používateľov. Tí môžu navzájom hodnotiť a komentovať anotácie na celom internete, nehladne na to, kto vlastní danú stránku a aké tam má povolenia. Príkladom môže byť situácia kde vydavateľ obsahu na video portáli YouTube zablokuje komentáre k svojmu videu. S týmto rozšírením sa môže rozbehnúť diskusia, tvorba poznámok a zvýrazňovanie bez obmedzení. Základné funkcie tohto nástroja sú zvýraznenie a anotácia. Zvýraznenie graficky rozlišuje vybraté prvky a pridá ich do zoznamu zvýraznení na stránke. Anotácia zahŕňa možnosť pridávania textových komentárov, obrázkov, odkazov a k tomu ešte značky, ktoré môžu slúžiť na triedenie a identifikáciu typu anotácie. Anotácie je možné pridávať verejne aj súkromne alebo do vlastne vytvorených skupín. Projekt ako celok vyzerá ako veľmi zaujímavé riešenie pre jednoduché anotácie, s ľahkou inštaláciou a intuitívnym použitím. Nevýhodou je nedostatok užívateľov, čo podkopáva predstavu tvorcov, kde internet je miesto plné interakcii ľudí pomocou anotácii.

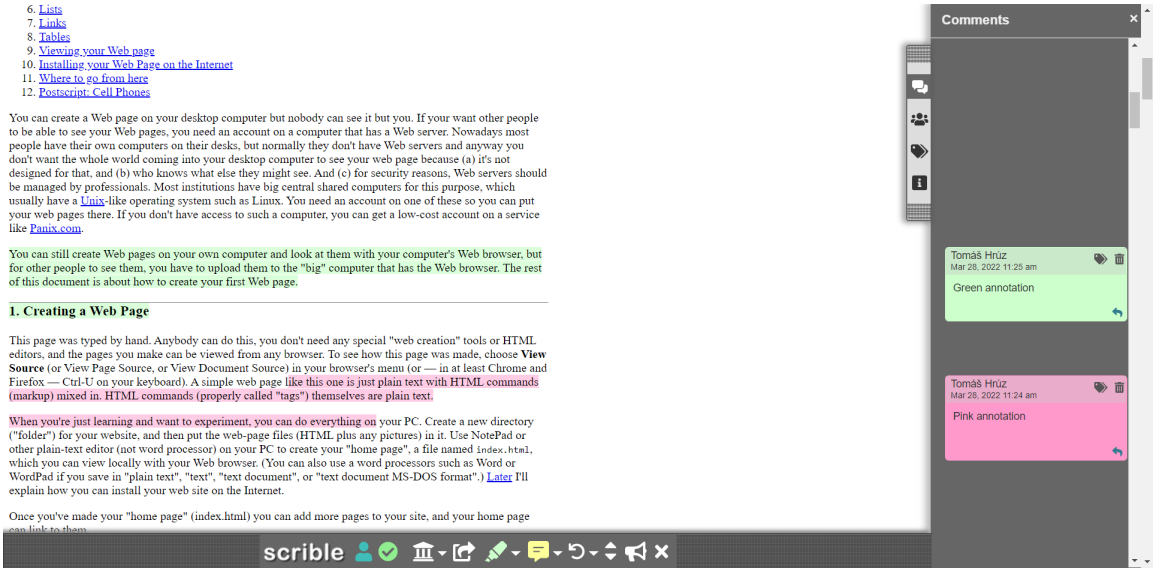


Obr. 3.1: Ukážka vrstvy nástroja Hypothesis nad webovou stránkou. Na ľavej strane obsah stránky so zvýraznenými prvkami. Na pravej strane panel s anotáciami od užívateľov a ovládacími tlačidlami

3.3 Scrible

Scrible[2] je webový anotačný nástroj so zameraním pre študentov a učiteľov. Dá sa použiť ako rozšírenie pre prehliadače Google Chrome alebo Microsoft Edge. Poskytuje viac plánov, niektoré z nich sú zadarmo ale sú aj platené verzie. Zdarma verzie sú podstatne okresané o funkcionalitu, kde je možné iba zvýrazňovanie textu štyrmi farbami, komentovanie, pridávanie značiek a využitie iba jednej knižnice dokumentov, čo na základne potreby anotácie stačí ale pre také prípady existujú aj lepšie nástroje. Platený plán pre študentov a učiteľov už ponúka oveľa zaujímavejšiu funkcionalitu. Tvorba viacerých knižníc dokumentov, zdieľanie knižníc, väčší výber zdrojov pre anotáciu, výber štýlu textu a veľa ďalších. Taktiež podporuje fungovanie s výukovými aplikáciami od Google, ktoré sa v Českej republike moc nevyužívajú ale môže to byť dobrý nástroj pre študentov z USA, pre ktorých to je primárne mierené.

[2] https://www.scrible.com/

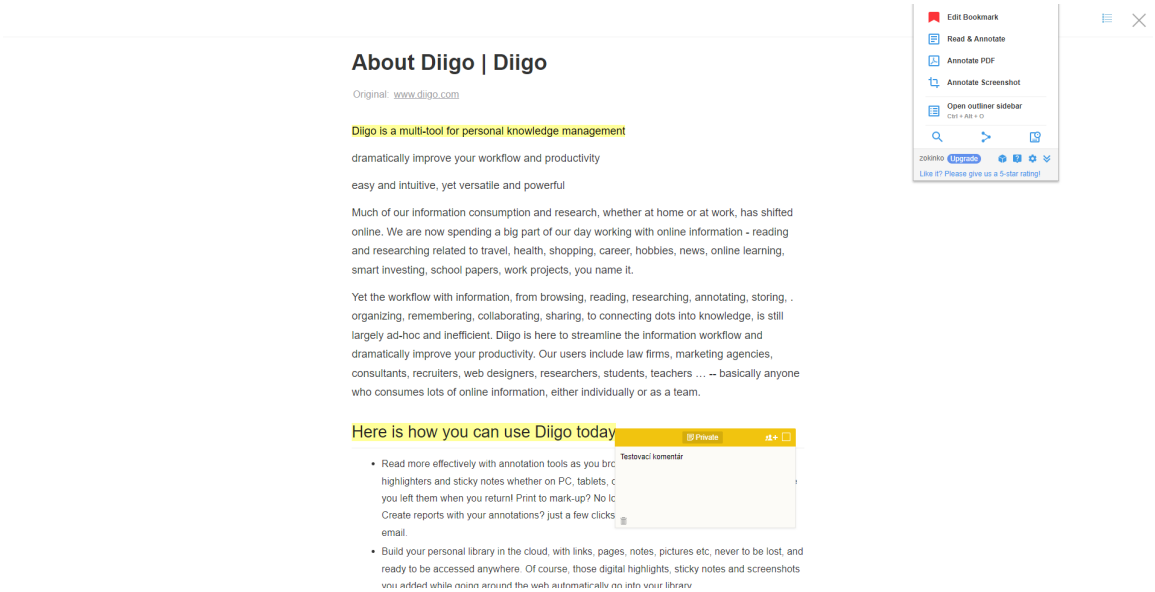


Obr. 3.2: Ukážka nástroja Scrible nad webovou stránkou. Na ľavej strane obsah stránky so zvýraznenými prvkami. Na pravej strane panel s farebne rozlýšenými anotáciami a ovládacími tlačidlami

3.4 Diigo

Diigo[3], čo je skratka pre "*Digest of Internet Information, Groups and Other stuff*" je rozšírenie do prehliadača Google Chrome na anotáciu dokumentov, alebo ako vravia tvorcovia je to nástroj na správu osobných vedomostí. Diigo ponúka verziu zdarma s obmedzeniami čo sa týka množstva anotácii a je prístupný iba s reklamami. Cenové plány sa pohybujú v rozsahu od 40 do 120 dolárov na rok. Ponúka veľké množstvo nástrojov. Vytváranie záložiek na webové stránky a dokumenty a zbierať ich na jednom mieste. Pridávanie tagov na kategorizáciu obsahu. Grafické zvýrazňovanie textu. Pridávanie poznámok priamo do obsahu stránky. Archivácia obsahu aj po tom čo pôvodný zdroj prestane existovať. Zdieľanie s ostatnými. Diigo nepracuje priamo nad webovou stránkou alebo dokumentom ale v prípade anotácie vytiahne obsah do dedikovaného rozhrania, kde je možné pracovať. Cieľom tohto projektu nie je žiadny špeciálny plán alebo zameranie ako pri niektorých iných nástrojoch. Tvorcovia chcú vytvoriť najefektívnejšie a najlepšie prostredie pre správu informácii a časť názvu "*Other stuff*" v ich prípade symbolizuje potrebu sa vždy zlepšovať a pridávať inovácie.

[3] https://www.diigo.com/

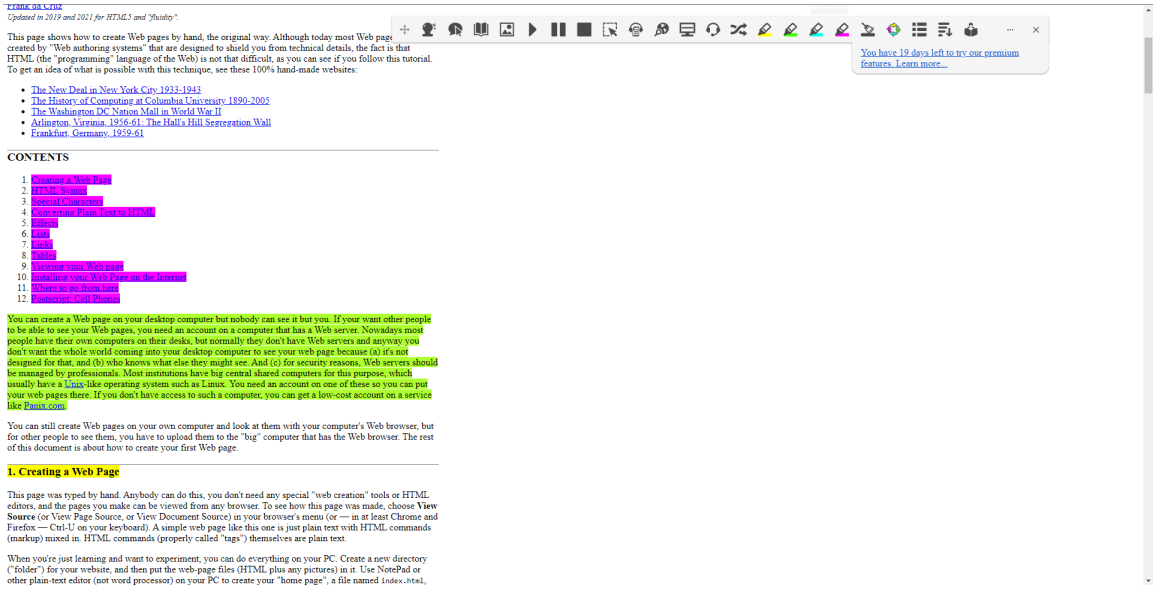


Obr. 3.3: Ukážka nástroja Diigo v jeho vlastnom rozhraní a s obsahom z anotovanej webovej stránky. V hornej časti je menu s možnosťami nástroja a priamov texte sú vyznačené časti a poznámka.

3.5 Read&Write for Google Chrome

Je to jednoduché rozšírenie pre Google Chrome od firmy TextHelp umožňujúce ľahkú a intuitívnu anotáciu dokumentov[4]. Zameranie nástroja je pre výuku a to pre študentov tak aj učiteľov ale dá sa využiť aj v komerčnom prostredí. Existuje len v platenej verzii, s výnimkou špeciálneho balíčku pre učiteľov, ktorý je zadarmo. Na vyskúšanie existuje 30 dňová skúšobná lehota. Aj napriek tomu, že neponúka verziu zdarma, tak môže byť zaujímavým riešením pre mnoho študentov, pre ktorých iný nástroj nebude možné používať. Read&Write si totiž zakladá svoju politiku na prístupnosti pre zdravotne znevýhodnené osoby. Jedná sa hlavne o zrakovo postihnutých ale aj mentálne postihnutých a ľudí s poruchami učenia. Na splnenie týchto cieľov používa prevod textu na reč pasáží alebo aj celých dokumentov. Opačný prístup, prevod reči na text pre diktovanie zadávaného textu. Doplňanie slov, oprava gramatiky a význam slov. Prepojenie s Google docs, odkiaľ je možné priamo púšťať nahrávky.

[4] https://www.texthelp.com/en-gb/products/read-and-write-education/



Obr. 3.4: Ukážka nástroja Read&Write nad webovou stránkou. V hornej časti sa nachádza panel s funkciami. V ľavej časti sú vyznačené časti textu a anotácia nad vyznačenou časťou.

3.6 Zhrnutie

Po otestovaní rôznych nástrojov sa ukázalo, že na trhu sa nachádza veľké množstvo nástrojov, z ktorých sa dá vyberať. Mnoho z nich má nejakú jedinečnú myšlienku alebo funkcionalitu, ktorou sa odlišuje od konkurencie a preto si nájdu svoje publikum. Je to napríklad zameranie pre zdravotne hendikepovaných ľudí, podpora funkcionality pre výuku na školách alebo myšlienka prepojenia internetu pomocou anotácii prístupných v celom internete. Preto aj napriek väčšiemu množstvu riešení, trh nie je zahltený a existuje miesto pre ďalšie originálne nápady. Veľa nástrojov je skrytých za poplatkami a to často vo forme predplatného, čo môže byť hlavne pre žiakov a študentov odrádzajúce, pričom práve oni tvoria veľké cieľové publikum. Na záver sa teda ukázalo niekoľko dôležitých faktorov. Je to hlavne možnosť farebného zvýrazňovania textu, pridávanie značiek a triedenie podľa týchto značiek, pridávanie komentárov k textu a originálna myšlienka, ktorá nástroj odlíši na trhu. Všetky testované nástroje boli prístupné ako rozšírenia do webových prehliadačov a to značí záujem o jednoduché a prístupné riešenia, integrované priamo v prehliadači a bez zložitejšej inštalácie.

Kapitola 4

Návrh riešenia

Táto kapitola sa najskôr pozrie na anotačný nástroj FitLayout. Táto bakalárska práca rozširuje FitLayout o novú komponentu a vylepšuje niektoré existujúce časti, preto je potrebné poznať ako všetko funguje a potom na tom budovať. Ďalej sa preberú požiadavky na novú funkcionalitu. To zahŕňa vysvetlenie potrebných častí, ktoré je potrebné implementovať. Nasleduje konkrétny návrh implementácie týchto požiadaviek ako po grafickej tak aj funkčnej stránke.

4.1 FitLayout

FitLayout[1][14] je aplikačný rámec, vytvorený docentom Radkom Burgetom, určený na tvorbu aplikácii a algoritmov, ktoré analyzujú webové dokumenty. Zjednodušuje úlohy spojené so získavaním vizuálnej podoby (rendering), modelovaním a analýzou. O rendering a segmentáciu obsahu sa stará aplikačná logika v jazyku Java. Je možná aj integrácia s aplikáciami, ktoré nie sú v Jave a to cez rozhranie príkazového riadku(CLI), alebo cez webové aplikačné rozhranie. Umožňuje spracovanie, segmentáciu, anotáciu, značkovanie a ďalšie nástroje pre zlepšenie pracovného toku. Dokumenty sú reprezentované, ukladané a dotazované vo formáte RDF.

4.1.1 Artefakty

Artefakty sú základom celej analýzy dokumentu. Reprezentujú každý produkt, ktorý je výsledkom analýzy ako je vykreslenie, segmentácia a ďalšie analytické operácie. Artefakt môže vytvárať služba artefaktov alebo je zdedený od rodičovského artefaktu. Vykreslená stránka sa potom skladá zo stromu artefaktov, kde stránka je koreňový artefakt a obsah stránky sú ďalšie artefakty vytvorené na základe rodičovských artefaktov.

- **Page (stránka)** - reprezentuje vykreslenú stránku ako strom boxov vytvorených backendom z DOM pôvodnej webovej stránky.
- **Area tree (strom oblastí)** - je to výsledok segmentácie stránky. Obsahuje strom vizuálnych oblastí vytvorených z pôvodných vizuálnych blokov stránky.
- **LogicalAreaTree (logická interpretácia stromu oblastí)** - je to logická interpretácia stromu oblastí. Tvoria ju logické oblasti skladajúce sa z grafických oblastí. Vzťahy medzi oblasťami môžu byť založené na ľubovoľnej spojitosti vyplývajúcej z kontextu segmentácie.

- **Text Chunk Set (súbor blokov textu)** – súbor textov extrahovaných z grafických oblastí pomocou algoritmu NER(Name entity recognition). Každý blok sa skladá z štvoruholníkovej oblasti a textového reťazca.

4.1.2 Služby

Sú to moduly, ktoré vytvárajú artefakty rôznych typov. Služby majú svoje jedinečné identifikátory, podľa ktorých sa potom môžu spustiť a obsahujú sadu vstupných argumentov. Najdôležitejšie služby sú:

- Backend pre vykreslovanie: **FitLayout.CSSBox, FitLayout.Puppeteer**
- Segmentačné metódy: **FitLayout.BCS, FitLayout.VIPS**
- Spracovanie stromu boxov: **FitLayout.VisualBoxTree**
- Spracovanie stromu oblastí: **Fitlayout.ApplyOperators**

4.1.3 Serializácia

FitLayout definuje ontologický model na vytváranie RDF popisov artefaktov, ktoré môžu byť serializované. Serializácia je možná okrem formátu RDF možná aj do formátov HTML,XML alebo PNG.

4.1.4 Repozitár

Repozitár slúži na ukladanie artefaktov. Pre dočasné uloženie medzi spracovaniami operácii sa využíva ukladanie v pamäti. Je možné aj ukladanie do RDF repozitára, do ktorého sa ukladajú RDF reprezentácie artefaktov a dotazuje sa naň jazykom SPARQL. Každý uložený artefakt má svoj jedinečný identifikátor IRI (*Internationalized Resource Identifier*), čo je internacionalizovaný identifikátor zdroja. IRI je reprezentovaný ako URL obsahujúce adresu webu alebo dokumentu na konci ktorého sa nachádza označenie artefaktu. Každý artefakt obsahuje aj typ, ktorý je tiež reprezentovaný ako IRI.

4.1.5 Rozhrania

Rozhranie príkazového riadka(CLI)

Prvá možnosť použitia nástroja FitLayout je cez príkazový riadok. Jeho inštalácia je možná dvoma spôsobmi, inštaláciou Java balíku alebo použitím docker kontajneru, čo je spojenie zdrojových kódov, knižníc a nadväzností potrebných pre spustenie programu. Pre interakciu využíva malú sadu príkazov. Príkazy:

- **RENDER** – vykreslí stránku
- **SEGMENT** – segmentuje stránku
- **EXPORT** – exportuje posledný vytvorený artefakt
- **USE** – otvorí repozitár pre načítanie a ukladanie artefaktov
- **LIST** – zoznam obsahu repozitára
- **LOAD** – načíta artefakt z repozitára
- **STORE** – uloží artefakt do repozitára

Java API

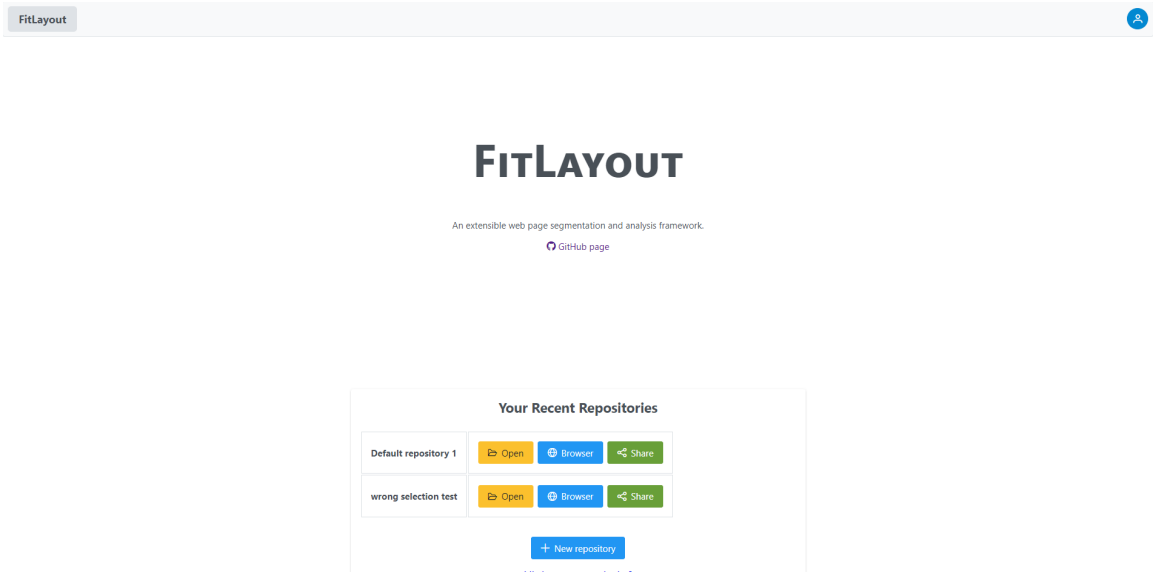
Java aplikačné rozhranie je určené na implementáciu do Java aplikácii. Ponúka funkcionalitu spomínanú v sekcii Služby 4.1.2.

REST API

Podobne ako aplikačné rozhranie v jazyku Java ponúka tú istú funkcionalitu. Poskytuje backend FitLayout webovej aplikácii.

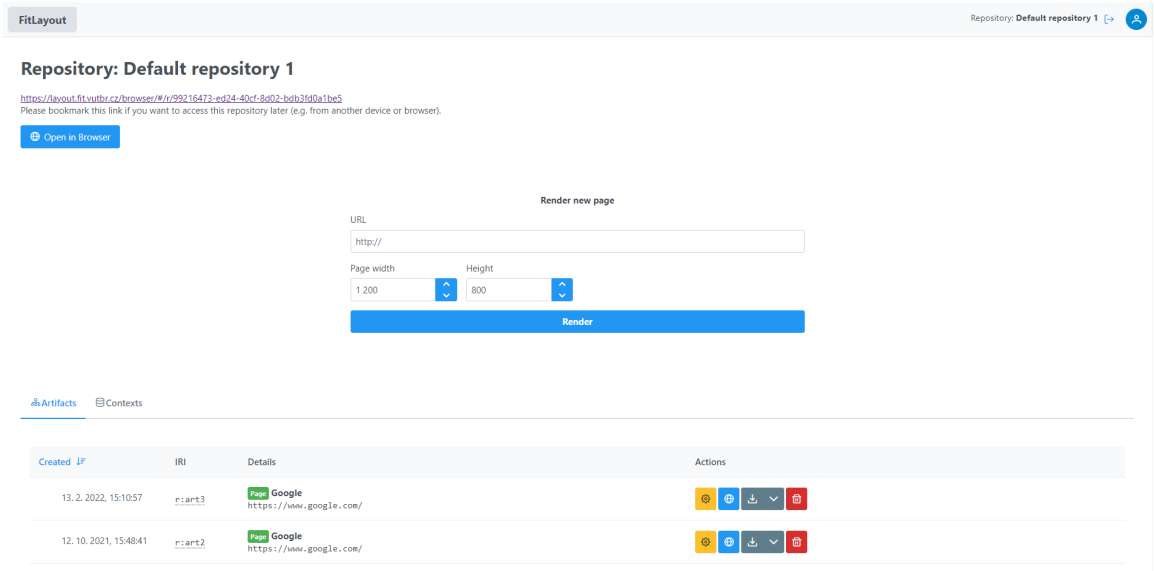
4.1.6 Webová aplikácia

Implementácia nástroja sa nachádza na adrese https://layout.fit.vutbr.cz/browser/#/ . Aplikácia bola vytvorená s využitím JavaScriptového rozhrania na tvorbu užívateľských rozhraní Vue.js. Užívateľa privíta jednoduché grafické rozhranie s možnosťou vytvárania nových repozitárov a so zoznamom už vytvorených repozitárov. Každý repozitár je možné otvoriť priamo v anotačnom rozhraní alebo sa dá pristupovať k metadátam repozitára. Repozitár je možné aj zdieľať pomocou URL adresy.



Obr. 4.1: Úvodná obrazovka aplikácie FitLayout, ktorá obsahuje vytvorené repozitáre

Po otvorení repozitára sa zobrazí rozhranie, v ktorom je možné pridávať nové webové stránky a dokumenty do repozitára. Stránky sa pridávajú pomocou URL adresy a pri pridávaní je možné zadať šírku a výšku vykreslenia. Po pridaní stránky do repozitára sa zobrazí na zozname artefaktov. S každou položkou sa dá robiť niekoľko operácii. Je možné zobraziť detaily artefaktov, exportovať v niekoľkých formátoch alebo sa artefakt z repozitára môže zmazať. Najdôležitejšou funkciou je však otvorenie stránky v anotačnom prostredí. Posledná súčasť tejto časti je pridávanie kontextov, kde je možné robiť rôzne nastavenia pre repozitár. Je možné napríklad pridávať nové tagy použité pre repozitár.



Obr. 4.2: Základné menu repozitára v aplikácii FitLayout

Sekcia anotácie môže vyzerať ako zahltená funkciami a potrebuje viac času na pochopenie, potom však prináša produktívne prostredie na anotáciu. V hornej časti sa nachádza menu kde je možné vyberať medzi typmi služieb z FitLayout API. Tieto typy zahŕňajú vykresľovanie, segmentáciu, spracovanie, text a vzťahy. Do výberu týchto služieb nemusí bežný užívateľ zasahovať, pretože všetko už je automaticky nastavené. Ak už by niekde potreboval zasahovať s najväčšou pravdepodobnosťou to bude sekcia vykresľovania, kde je možné pridávať a vykresľovať nové artefakty do repozitára.

V ľavej časti sa nachádza zoznam artefaktov z repozitára a tak isto ako na v hlavnom menu repozitára je možné ich zmazať, nie je ale možné vykonávať ďalšie úkony. Celý zoznam zaberá relatívne veľkú časť obrazovky, takže je možné ho skryť aby ďalšie časti mali väčší priestor.

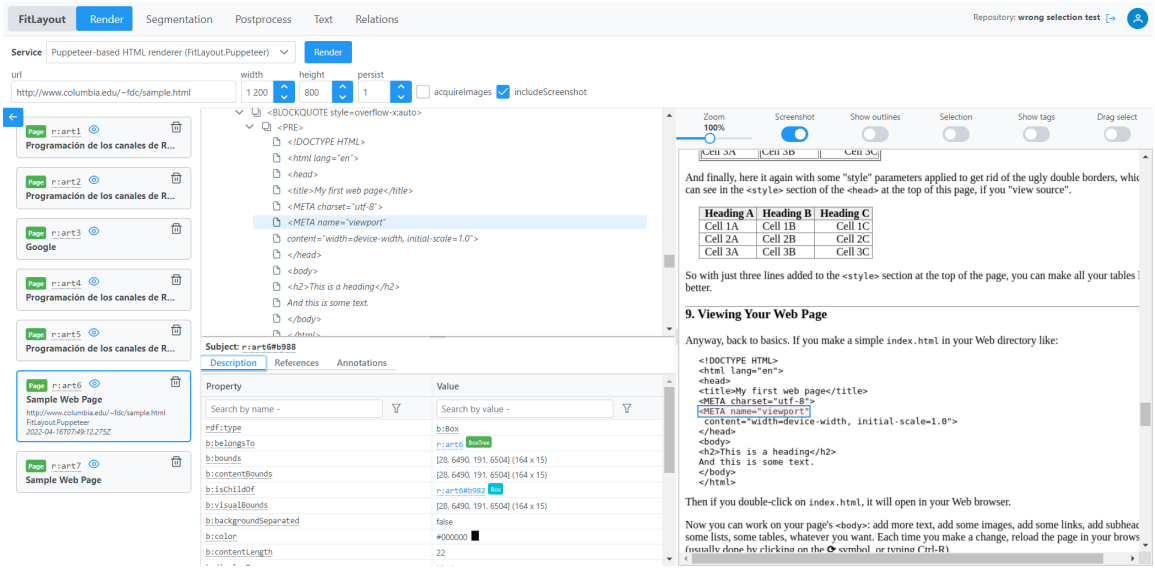
Hlavný pracovný priestor sa delí na dve časti. Pravá časť zobrazuje vykreslenú stránku, s ktorou sa dá interaktívne pracovať. Do obsahu je možné klikať a vyberať artefakty a pri prechode ponad tieto artefakty sa zvýrazňujú ich hranice. Nad týmto zobrazením sa nachádza panel s rôznymi možnosťami:

- **Zoom(priblíženie)** – je implementovaný pomocou posúvača a ovláda priblíženie vykreslenej stránky
- **Screenshot(Snímok obrazovky)** – prepínač, ktorý mení medzi pôvodným vykreslením a novým vykreslením vytvoreným službami FitLayout
- **Show outlines(Zobrazenie okrajov)** – prepínač zobrazí hranice boxov, ale len tenkou zelenou linkou aby čiary nerušili hlavný obsah
- **Selection (Výber)** – prepínač spúšťa funkciu výberu rôznych nezávislých boxov
- **Show tags (Zobrazenie značiek)** – prepínač pre vyfarbenie obsahu stránky podľa tagov

Ľavá časť zobrazuje všetky dôležité informácie k vykreslenému dokumentu. Zobrazuje jeho stromovú štruktúru a po kliknutí na box zobrazí konkrétny zakliknutý artefakt a rozbalí

jeho rodičovské komponenty. Toto previazanie funguje aj naopak, čiže po kliknutí na prvok v stromovej štruktúre sa zobrazí vyznačený box v dokumente Pod stromovou štruktúrou sú zobrazené metadáta artefaktu, ktoré sa podobne ako pozícia v stromovej štruktúre zobrazia po kliknutí na artefakt vo vykreslenom dokumente. Metadáta obsahujú tri sekcie:

- **Description(Popis)** – obsahuje rôzne metadáta k artefaktu ako napríklad súradnice jeho boxu, farbu textu, dĺžku obsahu, rodičovský artefakt a ďalšie. Informácie je možné aj filtrovať a vyhľadávať v nich
- **References(Odkazy)** – Obsahuje referencie k artefaktu
- **Annotations(Anotácie)** – obsahuje tagy(*značky*) a anotácie vybratého artefaktu



Obr. 4.3: Hlavné anotačné prostredie aplikácie FitLayout

4.2 Požiadavky

Súčasťou tejto práce je doplnenie funkcionality aplikácie FitLayout. Potrebné veci na implementáciu sú dva základné body:

- Prvá úlohaje vytvorenie modulu pre FitLayout, ktorý bude riešiť anotáciu väčších sekcii. Základná verzia aplikácie totiž podporuje iba anotáciu samostatných artefaktov. Takéto použitie je v reálnom použití potrebné ale rozhodne nie je dostačujúce. Spájanie artefaktov do skupín umožňuje napríklad spojenie tematicky podobných paragrafov do jedného logického celku, ktorý je možné anotovať. Práve takéto triedenie do sekcii je veľmi dôležité pre efektívnu prácu s informáciami.
- Druhá časť práce je vyriešenie samotnej anotácie. Základné rozhranie už je pripravené ale je potrebné vyriešiť prácu s anotáciami a značkami. Ich pridávanie, odoberanie a editáciu.

4.3 Návrh funkcionality

Nakoľko je aplikácia vytvorená s použitím Vue.js preto aj toto riešenie bude vytvorené pomocou tohto rámca. Podľa pravidiel tvorby Vue.js aplikácii by bolo vhodné vytvoriť samostatnú komponentu, ktorá bude zakomponovaná do aplikácie. Pre výber oblastí a pridávanie do sekcii to bude viac než vhodné aby sa kód logicky rozdelil od zvyšku aplikácie. Čo sa týka anotácie, tam nebude potrebné vytvoriť novú komponentu, pretože táto sekcia už je vytvorená v samostatnej komponente a je potrebné ju akurát prerobiť.

4.3.1 Anotačný panel

Upravenie anotačnej časti potrebuje implementáciu kompletnej práce s anotáciami a značkami. Pre značky to zahŕňa výber konkrétnej značky a jej pridanie do artefaktu. Pridané značky by mali byť viditeľné v nejakom zozname a každá samostatná značka musí mať možnosť na zmazanie. Editácia značiek nie je potrebná nakoľko značku stačí jednoducho zmazať a pridať novú.

Anotácie obsahujú viac funkcionality ako značky. Pri pridávaní je potrebné vybrať typ anotácie, pričom typ *"Label"* môže byť použitý raz a je to identifikačný popis. Ďalší typ anotácie je *"Comment"*, čo označuje klasický komentár a jedná sa o najbežnejší typ anotácie. K vybratému typu je ešte potrebné napísať krátky popis. A po zadaní týchto položiek je možné vytvoriť novú anotáciu. Tak isto ako pri značkách je vytvorené anotácie vidieť v zozname aby k ním užívateľ mohol pristupovať. Je potrebné mať možnosť ich zmazať a navyše ešte aj editovať popis.

4.3.2 Výber oblastí

Pri výbere oblastí je základná a najdôležitejšia funkcia výber oblasti, z ktorej sa budú zoskupovať artefakty. Toto je možné vyriešiť kliknutím na miesto kde má byť začiatok výberu a následným ťahaním až na koniec výberu, rovnako ako napríklad výber položiek v systéme Windows alebo Linux. Táto oblasť bude po výbere graficky vyznačená a bude ju možné pridať do stromu artefaktov. Pri vytvorení bude aj možnosť vybrať značku a "*Label*anotáciu. Po pridaní sa obnoví stromová štruktúra a bude obsahovať aj novo pridaný artefakt.

4.4 Grafický návrh

Tak isto ako funkcionalita aplikácie, je dôležité mať jednoduchý a zároveň zaujímavý vzhľad, pre prívetivú interakciu s užívateľom. Zároveň by mal vzhľad ladiť so zvyškom aplikácie či už farebnou paletou alebo vzhľadom komponent ako sú tlačidlá a textové polia. Aplikácia je v modrých farbách by nové časti mali byť modré alebo by s modrou aspoň mali ladiť.

4.4.1 Anotačný panel

Anotačný panel potrebuje dve hlavné tlačidlá 5.3. Jedno pre pridávanie anotácii a druhé pre pridávanie značiek, ale keďže je potrebné pri vytvorení v oboch prípadoch zadať parametre nemôže to byť iba jednoduché stlačenie tlačidla. Pre pridávanie parametrov sa po kliknutí na tlačidlo vyroluje panel s možnosťami na zadanie parametrov. Pre tag a typ anotácie sa môže použiť rozbaľovacia ponuka, pretože tie budú jasne dané. Pre popis anotácie sa použije textové pole. Na odoslanie požiadavku stačí obyčajné tlačidlo. Tlačidlá budú v tvare

zaobleného štvorca, čo v dnešnej dobe vyzerá elegantne a šetrí to miesto oproti klasickým pozdĺžnym tlačidlám. Do tak malých tlačidiel sa nezmestí slovný popis a preto sa využijú ikony, ktoré budú značiť funkcionalitu tlačidla. Ikony budú v bielej farbe čo spolu s modrým tlačidlom vytvorí pekný kontrast. Pre prípadné nejasnosti pri použití bude pri prechode myšou nad tlačidlom zobrazený pomocník. Pridané anotácie budú zobrazené pod sebou vo voľnom priestore anotačného panelu a každá anotácia bude obsahovať typ anotácie, jej popis a tlačidlá na editáciu a mazanie s modro-bielym motívom a ikonou. Tagy budú zobrazené nad anotáciami a všetky tagy budú súčasťou jednej položky v zozname. Každý tag bude mať svoje tlačidlo na odstránenie.

4.4.2 Výber oblastí

Výber oblastí by mal jasne ukazovať, ktoré oblasti sú vo výbere, takže sa bude jednať o priehľadný štvoruholník zobrazený nad obsahom s jemnou modrou farbou aby ladil s aplikáciou. Komponenty na pridanie by mohli byť zobrazené až po vybratí oblasti, čo by značilo, že pokiaľ nie je vybraná žiadna oblasť nie je možné nič pridať. Potrebné komponenty sú textové pole pre zadanie popisu, rozbaľovacia ponuka pre tag a tlačidlo na pridanie oblasti. Tieto komponenty by mali plávať nad obsahom aby boli prístupné všade v zobrazenej stránke. Po pridaní oblasti by mali pridávacie komponenty zmiznúť. Pre spustenie tejto funkcionality by sa mal pridať ešte prepínač na panel s možnosťami, ktorý bude túto možnosť spúšťať.

Kapitola 5

Implementácia

V tejto kapitole je popísaná konkrétna implementácia navrhovaných častí. Sú predstavené použité technológie a ich využitie pri tvorbe práce. Pretože o backend sa vo svojej aplikácii stará docent Radek Burget, implementácia je zameraná na frontend. Z vytvorených komponent sa potom volajú už pripravené funkcie.

5.1 Výber technológii

Výber technológii bol obmedzený na už použité technológie v pôvodnej aplikácii FitLayout a preto bolo žiadané využiť tie technológie, ktoré už boli použité.

5.1.1 GitHub

Ako verziovací systém bol použitý GitHub práve z dôvodu už pôvodného použitia pri tvorbe FitLayout. Celý projekt sa nachádza na odkaze https://github.com/FitLayout , kde je vytvorená organizácia pre celý projekt. V tejto organizácii sa nachádza repozitár "*PageView*ä ten obsahuje webovú aplikáciu. Z tohto repozitára sa vytvorila samostatná vetva na pridávanie novej funkcionality. Pre fungovanie aplikácie je ešte potrebné spustiť backend, ktorý sa nachádza v repozitári "*FitLayoutWeb*"v rovnakej organizácii.

5.1.2 Prvé spustenie

Webová aplikácia už pôvodne fungovala online ale pre potreby testovania bolo potrebné spustiť lokálnu verziu aplikácie. Spustenie zahŕňalo inštaláciu niekoľkých technológii. Pre spustenie backendu bolo potrebné nainštalovať Java SDK vo verzii 11 pre správnu kompatibilitu. V repozitári sa nachádza nepreložená verzia backendu ale je možné vytvoriť spustiteľný .jar balíček, ktorý zapne backend server starajúci sa o služby aplikácie FitLayout. Spustenie serveru na porte 8080 je možné príkazom:

```
java -jar fitlayout-web-microbundle.jar -port 8080
```

Aplikácia je písaná v jazyku Vue.js a využíva špeciálne komponenty, ktoré ponúka knižnica PrimeVue. Pre tieto technológie nie je potrebné inštalovať nič špeciálne ale je potrebné nainštalovať Node.js a NPM, ktoré sa využívajú pre všeobecné spúšťanie JavaScript aplikácii. Pomocou týchto služieb je potom možné zapnúť aplikáciu príkazom:

```
npm run serve
```

Po úspešnej inštalácii nástrojov a ich spustení by mala aplikácia bežať na localhost adrese.

5.1.3 Vue.js

Vue.js použité v tejto aplikácii je verzie 3. To indikuje, že by pre komponenty mal byť použitý spôsob zápisu Composition API. Pôvodné komponenty však využívajú starší model zápisu Options API. Aj keď je toto riešenie zastaralé je v poriadku ho používať ale mal by sa klásť dôraz na to aby sa všade písalo jedným spôsobom bez miešania.

5.1.4 PrimeVue

Komponenty použité v aplikácii využívajú knižnicu PrimeVue, ktorá uľahčuje tvorbu aplikácie, pretože vývojár nemusí vyvárať niektoré komponenty od základu, prípadne nemusí vytvárať sám všetky štýly. PrimeVue poskytuje dostatočne veľké množstvo tém pre každého a malo by poskytovať všetky základné komponenty. Aplikácia obsahuje často používané komponenty, ktoré je dobré pred písaním ďalšieho kódu poznať:

- **Button** – sa používa ako klasické tlačidlo, ktoré využíva moderný vzhľad a umožňuje jednoduché používanie ikon namiesto textu
- **Slider** – klasický posúvač s rozšírenými možnosťami a moderným vzhľadom
- **InputText** – klasické textové pole s moderným vzhľadom. Podporuje napríklad pridávanie ikon priamo do poľa
- **InputSwitch** – prepínač pre výber pravdivostných hodnôt s moderným vzhľadom a animáciou
- **Dropdown** – rozbaľovacia ponuka s podporou vyhľadávania, zoskupovania, pridávania ikon a moderným vzhľadom
- **OverlayPanel** – komponent zaobaľujúci ľubovoľný obsah. Je pripojený na svoj rodičovský komponent a zobrazuje sa nad pôvodným obsahom

5.2 Štruktúra kódu

Hlavnú časť aplikácie tvorí súbor App.vue. Je to hlavný komponent aplikácie a obsahuje smerovač pohľadov medzi ktorými užívateľ prepína pri pohybe v aplikácii. Tento komponent je vytvorený na začiatku aplikácie zo skriptu main.js. Následne prepínané pohľady sú v zložke Views. Tieto pohľady sú podľa princípov tvorby Vue.js aplikácii tvorené množinou komponent, ktoré sa do seba postupne zanárajú. Pohľady v aplikácii FitLayout:

- AdminRepos.vue
- AdminView.vue
- BrowserView.vue
- Home.vue
- PageDetailView.vue
- RepositoryContentView.vue
- RepositoryView.vue

Každý komponent reprezentuje jeden pohľad v aplikácii. Podľa rozdelenia v sekcii webová aplikácia Home.vue reprezentuje domovskú obrazovku, ktorá umožňuje výber a vytváranie repozitárov. Po otvorení konkrétneho repozitára sa zobrazí pohľad vytvorený súborom RepositoryContentView.vue. Pre túto prácu je podstatný BrowserView.vue reprezentujúci hlavné anotačné rozhranie konkrétnych artefaktov. V tomto pohľade sa nachádzajú komponenty, tvoriace výberové menu služieb v hornej časti obrazovky a zvyšok pohľadu zaberá vlastný komponent PageView.vue obsahujúci všetky potrebné komponenty pre implementáciu novej funkcionality.

5.2.1 PageView

Jeden z pohľadov, v aplikácii je PageView.vue. Je to najdôležitejší pohľad z hľadiska anotácie. Implementuje rozhranie anotácie, v ktorom je otvorený konkrétny artefakt z repozitára. Skladá sa z vlastných ale aj originálnych PrimeVue komponent. Skladá sa z originálnych PrimeVue komponent ale aj vlastných vytvorených komponent. Vlastné komponenty PageView:

- Page.vue
- Iri.vue
- ValueInfo.vue
- AnnotationPanel.vue

Iri.vue a ValueInfo.vue nie sú podstatné pre ďalšiu implementáciu, preto ich netreba bližšie predstaviť. AnnotationPanel.vue zobrazuje a stará sa o anotáciu artefaktov a bude v ňom potrebné implementovať požadované funkcie. Page.vue vykonáva zobrazenie webovej stránky alebo dokumentu a vyžaduje pridanie funkcie výberu oblasti a jej pridanie do stromu artefaktov. Toto chovanie bude ale podľa návrhu pridané do samostatnej novo vytvorenej komponenty aby sa udržala prehľadnosť v kóde.

5.2.2 Page

Z hľadiska HTML štruktúry sa jedná o jednoduchý komponent obsahujúci zobrazené boxy artefaktov. Komplexnosť sa nachádza v časti JavaScriptu, ktorá sa stará o správne vykreslenie boxov do komponenty. Tieto boxy je potrebné nejakým spôsobom označiť a pridať do skupiny, ktorá bude vložená do stromu artefaktov ako nový artefakt. Kvôli spomínanej prehľadnosti bude dobré nechať tomuto komponentu funkciu vykresľovania a nové funkcie pridať do nového komponentu.

5.2.3 Selection

Je to nový komponent vytvorený pre potreby výberu boxov a ich pridávania do stromu artefaktov ako jeden nový artefakt. Komponent by mal byť nad pôvodným komponentom zobrazených boxov. Tie by mali byť stále viditeľné ale zároveň by mali byť prístupné funkcie tohto komponentu. Pre takéto riešenie je vhodné použitie klasického elementu `<div>` z HTML. Tento element bude prekrývať celý komponent ako priesvitná vrstva, s ktorou je možné interagovať. Interakciu zahŕňa implementácia základných udalostí s myšou, ktoré docielia navrhovaný výber oblasti ťahaním myši. Naviazanie metód na udalosti je vo Vue.js veľmi jednoduché a zapisuje sa priamo v HTML kóde ako atribút elementu.

Spôsob zápisu: `v-on:<názov udalosti>=«názov metódy>"`

Výber oblasti bude reprezentovaný ako štvoruholník ťahaný pohybom myši a po uskutočnení výberu ostane zobrazený okolo hraníc boxov, ktoré boli v ňom obsiahnuté a uloží si pole vybratých boxov do pamäti. Výberový štvoruholník bude implementovaný elementom `<div>`.

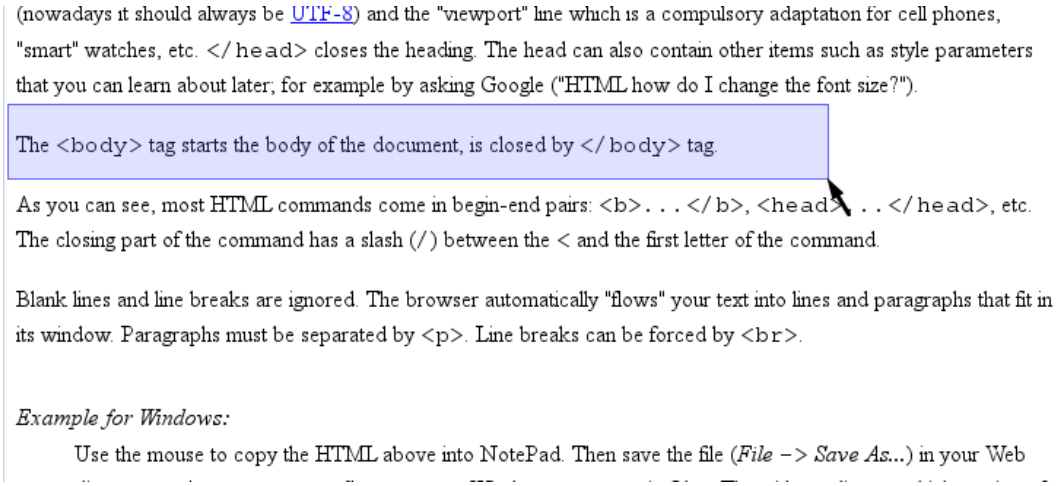
Pre správne chovanie je nový komponent použitý v komponente PageView, kde je použitý komponent Page a ten obaľuje volanie komponentu Selection.

Stlačenie tlačidla myši

Stlačením tlačidla myši začína súbor operácii potrebných pre vytvorenie skupiny boxov. Najprv sa vytvorí nový element `<div>` pomocou funkcie `document.createElement('div')`. Dynamicky vytvorí nový element, ktorý ešte nemá vizuálnu podobu ani žiadne vlastnosti a tie mu treba nastaviť. Príkazom `setAttribute(<typ atribútu>, <hodnota atribútu>)` sa nastavujú potrebné atribúty. Elementu sa nastaví identifikátor aby sa k nemu dalo pristupovať. Z udalosti sa zistia súradnice X a Y pre vytvorenie počiatočných súradníc. Ďalší dôležitý atribút je štýl. Jeho začiatočné súradnice sú známe z udalosti a šírka a výška sa zatiaľ nastaví na 0. Farba divu je modrá aby ladila so zvyškom aplikácie a pozadie je mierne priesvitné aby bolo vidieť vyberaný obsah. Pre správne zobrazenie je pozícia nastavená na absolútnu a index osi Z je na vyššej hodnote ako pôvodný komponent. To zaručí že bude výberový štvoruholník bude zobrazený nad pôvodným obsahom.

Pohyb myši

Pri tejto udalosti sa zisťujú súradnice aktuálnej polohy kurzoru. Ich získavanie sa vykonáva iba v prípade, že je stlačené ľavé tlačidlo myši, v opačnom prípade sa nevykonáva nič. Od aktuálnej pozície sa odčítajú počiatočné hodnoty z čoho sa zistí výška a šírka výberového elementu.



Obr. 5.1: Výber oblasti ťahaním myši

Pustenie tlačidla myši

Po ukončení požadovaného výberu užívateľ pustí ľavé tlačidlo myši a tým ukončí výber. V tomto stave už výberový element nie je potrebný a odstráni sa. Pomocou fun-

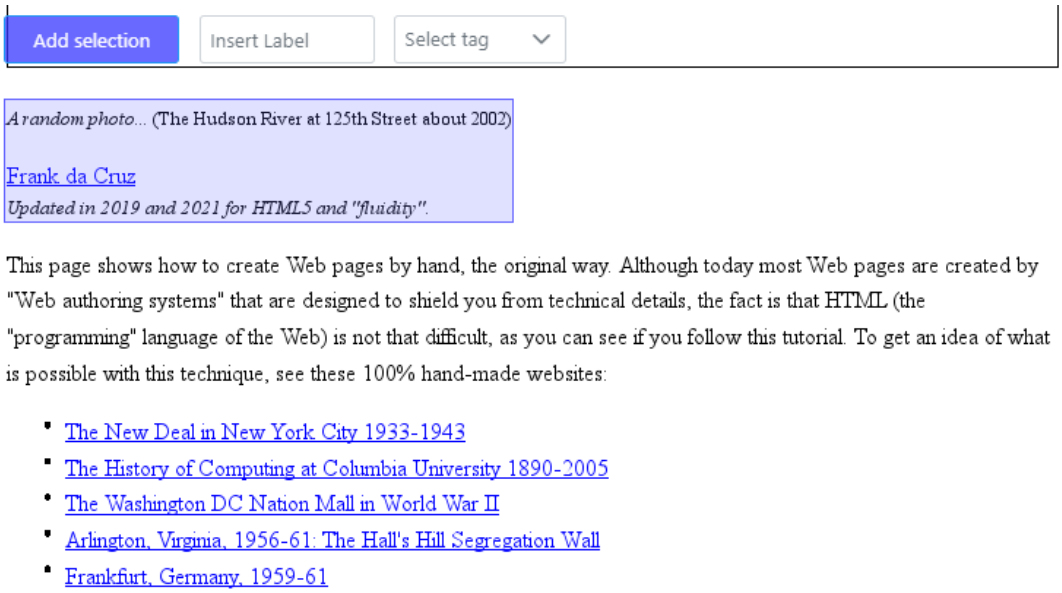
kcie `document.getElementById('<identifikátor elementu>')` sa nájde daný element a zavolaním metódy `remove()` sa odstráni zo stromovej štruktúry dokumentu. Uložia sa súradnice udalosti a spolu s počiatočnými súradnicami výberu je možné vypočítať súradnice nového elementu `<div>`, ktorý bude ohraničovať len vybrané boxy a to v prípade, že sa nachádzali celou svojou plochou vo výbere.

Komponent Selection ale sám o sebe nevidí žiadne boxy. Zoznam boxov obsahuje komponent PageView a musí ich posunúť ďalej. V tomto prípade je to vyriešené atribútom "*props*", čo je najjednoduchší spôsob zdieľania dát medzi komponentami. Zápis je následovný:

`<Selection :pageRectAreas="rectangles»</Selection>`

Premenná "*rectangles*"z komponentu PageView je uložená do premennej "*pageRectAreas*", ktorá je súčasťou sekcie "*props*"v komponente Selection.

Každý box zo zoznamu sa prejde a skontroluje, či sa nachádza v hraniciach výberu a prepočítajú sa hodnoty pre nový ohraničujúci štvoruholník. Ak sa nachádzal aspoň jeden box vo výbere, tak sa do pamäti uloží zoznam boxov, vykreslí sa nový `<div>` ohraničujúci vybrané boxy a zobrazia sa komponenty potrebné pre vytvorenie nového artefaktu.



Obr. 5.2: Vybraná oblasť zarovnaná na hranice boxov a zobrazené komponenty pre pridanie oblasti

Opustenie myši mimo komponent

Táto udalosť sa stará iba o situácie, že užívateľ by počas výberu opustil pohľad na vykreslený dokument. V takom prípade sa odstráni element `<div>` vytvorený pri kliknutí myši rovnako ako pri udalosti "*Pustenie myši*".

Vytvorenia nového artefaktu

Prvok by mal obsahovať ďalšie komponenty pre vykonanie potrebnej funkcionality. V tomto prípade je vhodné použiť komponenty z knižnice PrimeVue. Pre pridávanie popisu oblasti je použitý komponent `<InputText>` s prednastaveným textom, ktorý značí čo má byť do textového poľa vložené. To eliminuje potrebu pridávať ďalší komponent, ktorý by popisoval

čo je potrebné zadať. Pre pridávanie značiek je podľa návrhu využitá rozbaľovacia ponuka. PrimeVue na to ponúka komponent `<Dropdown>` a podobne ako popis novej oblasti aj tento komponent umožňuje zobrazovať text, kde sa píše čo sa v ponuke vyberá.

Popis artefaktu a výber značky je voliteľný a v prípade, že sa nezadá popis bude do artefaktu vložený predložený popis "*no label*". Ak nebude vybraná žiadna značka nebude vložené nič. Stlačením tlačidla na pridanie sa zisti ešte IRI rodičovského artefaktu z atribútu `box.isChildOf._iri` a pôvodného artefaktu z atribútu `box.belongsTo._iri`, z ktorého vznikne nový artefakt. Tieto údaje sú prístupné v každom boxe a v tomto prípade sa dostanú z prvého vybraného boxu. IRI nového artefaktu musí byť jedinečné a preto sa nemôže použiť IRI pôvodného artefaktu. Nové IRI bude tvorené teda pôvodným IRI oddeleným znakom '#' a nasledovaným spojením identifikátorov všetkých vybraných boxov.

Príklad nového IRI artefaktu:
`http://fitlayout.github.io/resource/art25#b407b408b409b410b411b412b413b414`

Po získaní všetkých potrebných informácii sa zavolá funkcia z backendu na vytvorenie novej oblasti `rdfUtil.createSuperArea(artIri, parent, children, data)` pričom argumenty funkcie znamenajú:

- **artIri** – IRI pôvodného artefaktu
- **parent** – IRI rodičovského artefaktu
- **children** – pole boxov, ktoré sa nachádzajú vo výbere
- **data** – obsahuje súradnice a IRI nového artefaktu, a prípadnú značku alebo popis

Všetko sa ukončí obnovením pohľadu na strom artefaktov, vymaže sa výberový štvoruholník a pridávacie komponenty prestanú byť viditeľné.

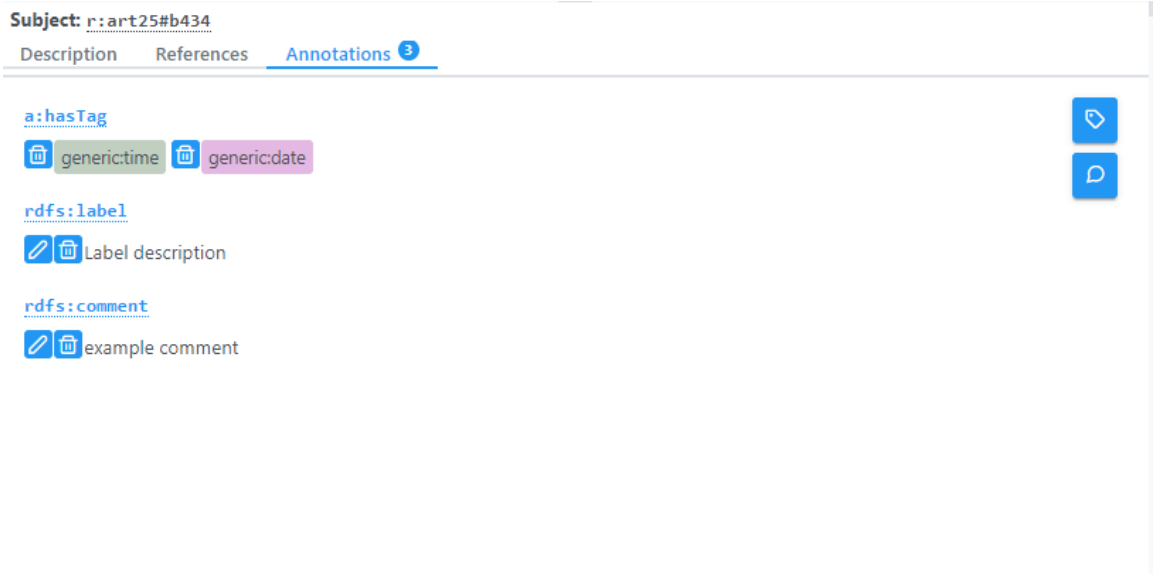
5.2.4 AnnotationPanel

Tento komponent je už vytvorený a je potrebné už len dorobiť možnosti interakcie so značkami a anotáciami. Pridané značky a anotácie je potrebné rozlišovať, pretože zatiaľ čo značky je potrebné len zmazať pri anotáciách ich navyše treba aj upravovať. Pre rozlíšenie je použitá vlastná vytvorená funkcia `compareAnnotationOrTag(item)`, ktorej parameter je položka vykresleného zoznamu. Vo funkcii sa overí či sa jedná o značku na základe vedomosti, že má na konci názvu reťazec "*hasTag*".

Porovnanie musí ovplyvniť vykreslenie položiek zoznamu, takže sa musí volať pri vykreslení stránky. Vue.js pre takéto prípady ponúka riešenie. Funkcia bude volaná priamo v HTML prvku a to z parametru „v-if“, ktorému bude priradená spomínaná funkcia. Toto sa nazýva podmienené vykresľovanie a bude vykonané iba v prípade, že sa splní podmienka. Nasledujúci prvok bude obsahovať parameter „v-else“ a vykoná sa ak nebude splnená podmienka v predchádzajúcom prvku.

Týmto spôsobom sa položka vykreslí ako položka značiek, kde každá značka bude mať svoje tlačidlo na vymazanie alebo sa vykreslí položka anotácie, kde bude jedno tlačidlo na editáciu a mazanie. Pri vykreslení sa tlačidlám priradí prislúchajúca položka, ktorá bude zmazaná alebo editovaná pri stlačení. Zmazaním anotácie sa zavolá funkcia backendu `apiClient.deleteValue(this.subjectIri, item.iri, this.artifactIri)`, ktorá odstráni vybranú položku. Podobne to je aj pre značku kde sa volá funkcia `apiClient.deleteTag(this.subjectIri, item, this.artifactIri)`. Editácia anotácie má krok navyše v tom, že sa po kliknutí tlačidla vyroluje komponent `<OverlayPanel>` obsahujúci

textové pole obsahujúce pôvodný obsah a tlačidlo na potvrdenie zmeny. Po stlačení tlačidla sa zavolá funkcia mazania a vytvorí sa nová anotácia funkciou `apiClient.addValue( this.subjectIri, item.iri, this.labelEditText, this.artifactIri)`.



Obr. 5.3: Značky a anotácie s tlačidlami na editáciu a mazanie

Kapitola 6

Testovanie

V tejto kapitole sú vysvetlené spôsoby testovania a je zhodnotená výsledná implementácia modulov vo webovej aplikácii FitLayout.

6.1 Manuálne testovanie

Na testovanie aplikácie neboli použité žiadne automatizované testy. Všetko bolo testované manuálne a to v dvoch režimoch. Prvý režim zahŕňa testovanie lokálne na jednom počítači, ktorý mal spustený backend z Java balíčku a stránku pustenú na lokálnom serveri. V tomto režime bola aplikácia testovaná priebežne počas celého vývoja na súbore vstupných stránok. Druhý režim prebiehal na produkčnej verzii aplikácie, ktorá sa nachádza online. V tomto prostredí testovanie prebiehalo po väčších dokončených celkoch po spojení hlavnej a vývojovej vetvy na portáli GitHub.

6.2 Dôležité položky testovania

Správne fungovanie zahŕňalo dva hlavné faktory. Prvým z nich bolo správne vykreslenie potrebných komponent. Tlačidlá, textové polia a ďalšie komponenty by mali byť vykreslené tak ako je od nich vyžadované. To znamená že nebudú vykreslené krivo, na úplne inom mieste alebo nebudú vykreslené vôbec.

Ďalší faktor súvisí s výberom oblastí. Ten by mal fungovať správne nad každým dokumentom alebo stránkou. V praxi to znamená, že pokiaľ užívateľ vyberie množinu boxov tak prvok označujúci skupinu by mal byť vykreslený okolo všetkých vybratých boxov a to na hraniciach najkrajnejších boxov. Pravidlá výberu platia aj pre zoznam boxov, ktorý sa nachádza v hraniciach výberu. Boxy, ktoré sa pridávajú do nového artefaktu musia byť naozaj všetky z vybranej oblasti a nesmie žiadny chýbať.

6.3 Záverečné poznatky testovania

Záverečné testovanie po hotovej implementácii prebiehalo aj v online aj v lokálnej verzii s množinou testovacích stránok nad ktorými bol vykonaný správny výber novej oblasti a pridávanie značiek a anotácii. Samotný výber a pridávanie artefaktov a aj anotácie fungujú dobre aj napriek tomu, že počas vývoja mali chybné chovanie, ktoré ale už nie je prítomné. V riešení sa však nachádzajú dve chyby, ktoré súvisia s vykreslením. Jedna chyba je v komponente Selection, kde komponenty pre pridávanie oblastí majú plávať nad dokumentom

aby boli vždy viditeľné. V prípade použitia priblíženia však ostanú v hornej časti vykresleného dokumentu až do momentu obnovenia aplikácie. Druhý problém súvisí s volaním backend funkcie na obnovu pohľadu, ktorý v ojedinelých prípadoch nevykreslí komponenty aplikácie s aktuálnymi dátami.

Kapitola 7

Záver

Cieľom tejto práce bolo zoznámiť sa s tvorbou webových aplikácii, otestovať a porovnať existujúce riešenia pre anotáciu webových dokumentov. S týmito vedomosťami bolo následne potrebné zoznámiť sa s experimentálnym nástrojom FitLayout a implementovať výber oblastí spolu s ich pridávaním do stromovej štruktúry dokumentu a navyše k tomu dokončiť už pôvodné riešenie anotácie, ktorému chýbala potrebná funkcionalita ako je editácia, mazanie a celkové zaobchádzanie s anotovanými položkami.

Pri zoznámení sa s nástrojmi pre webové aplikácie bolo v tejto práci zistené, že progresívny JavaScript framework Vue.js je dominantný skoro vo všetkých ohľadoch. Toto bolo priaznivé zistenie nakoľko aplikácia FitLayout využíva práve Vue.js. Testovanie a porovnanie konkurenčných nástrojov pre anotáciu webových dokumentov ukázalo, že na trhu sa nachádza veľké množstvo riešení. Väčšina však prinášala jedinečný pohľad na vec a tým si zaručovali pozíciu na trhu a boli dôkazom toho, že aj pri veľkej konkurencii je možné preraziť.

Samotný návrh a implementácia modulu do nástroja FitLayout splnil požiadavky zadania. Použitý bol framework Vue.js vo verzii 3 spoločne s knižnicou komponent rozhrania PrimeVue. Prvý bod zadania bolo vytváranie nových oblastí v stromovej štruktúre dokumentov. Riešením bol samostatný komponent, ktorý dokázal označiť položky dokumentov a pridať ich ako nový artefakt. Druhá časť implementácie upravila už vytvorený komponent a pridáva kompletnú funkcionalitu práce s anotáciami ako je pridávanie, mazanie a editácia.

Finálne riešenie bolo testované na súbore dokumentov, kde sa ukázal správnosť implementácie s drobnými grafickými chybami, ktoré bude možné upraviť v budúcnosti. V práci by niekto mohol ďalej pridávať funkcionalitu, ktorá ďalej podporí triedenie a spracovanie informácii. Zaujímavá by bola podpora zvukového predčítania pre zrakovo postihnutých ľudí, prípadne iná pomocná funkcia.

Tvorba práce ma bližšie zoznámila s webovými technológiami a to hlavne s Vue.js. Tento framework ma veľmi zaujal svojou syntaxou, ktorá je veľmi efektívna a aj s pár riadkami kódu dokáže viac ako konkurenčné technológie. Zároveň s jeho stúpajúcou popularitou vyzerá, že tu bude ešte dlho a bude sa stále vylepšovať, čo z Vue.js zaujímavú voľbu pre budúci kariérny postup.

Literatúra

[1] Burget, R. *FitLayout/2 - Web Page Analysis Framework* [online]. FitLayout, október 2021 [cit. 2022-04-19]. Dostupné z: https://github.com/FitLayout/FitLayout/wiki.

[2] Codecademy Team. *What is Node?* [online]. Codeacademy [cit. 2022-04-19]. Dostupné z: https://www.codecademy.com/article/what-is-node.

[3] Codesido, I. *What is front-end development?* [online]. The Guardian, september 2009 [cit. 2022-04-19]. Dostupné z: https://www.theguardian.com/help/insideguardian/2009/sep/28/blogpost.

[4] Eubanks, B. *Java na maximum.* Vyd. 1. Brno: Computer Press, 2006. ISBN 80-251-1111-3.

[5] Flanagan, D. *JavaScript : kompletní průvodce.* 2. aktualizované vydání. Praha: Computer Press, 2002. ISBN 80-7226-626-8.

[6] Gregersen, E. et al. *Markup language* [online]. Britannica, august 2020 [cit. 2022-04-19]. Dostupné z: https://www.britannica.com/technology/markup-language.

[7] Harms, D. D. *Začínáme programovat v jazyce Python.* 1. vyd. Brno: Computer Press, 2003. ISBN 80-7226-799-X.

[8] Herko, L. *Frontend vs Backend: v čom je rozdiel?* [online]. Learn2Code, máj 2017 [cit. 2022-04-19]. Dostupné z: https://www.learn2code.sk/blog/front-end-vs-back-end.

[9] Koďousková, B. *Co je Angular, v čem je jiný než AngularJS a proč ho použít?* [online]. Rascasone, apríl 2021 [cit. 2022-04-19]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-angular-angularjs.

[10] Macrae, C. *Vue.js : up and running : building accessible and performant web apps.* First Edition. Sebastopol: O'Reilly, 2018. ISBN 978-1-491-99724-6.

[11] mDevelopers Team. *Frontend vs. Backend - What is what?* [online]. mDevelopers, december 2021 [cit. 2022-04-19]. Dostupné z: https://mdevelopers.com/blog/frontend-vs-backend-what-is-what-.

[12] MDN contributors. *CSS basics* [online]. MDN, apríl 2022 [cit. 2022-04-19]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS.

[13] MDN CONTRIBUTORS. *HTML: HyperText Markup Language* [online]. MDN, február 2022 [cit. 2022-04-19]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML.

[14] MILICKA, M. a BURGET, R. Information Extraction from Web Sources Based on Multi-aspect Content Analysis. In: GANDON, F., CABRIO, E., STANKOVIC, M. a ZIMMERMANN, A., ed. *Semantic Web Evaluation Challenges*. Cham: Springer International Publishing, 2015, s. 81–92. ISBN 978-3-319-25518-7.

[15] OBERLEHNER, M. *Vue 3 Composition API vs. Options API* [online]. Markus Oberlehner, júl 2021 [cit. 2022-04-19]. Dostupné z: https://markus.oberlehner.net/blog/vue-3-composition-api-vs-options-api/.

[16] ONTOTEXT TEAM. *What is SPARQL?* [online]. Ontotext USA [cit. 2022-04-19]. Dostupné z: https://www.ontotext.com/knowledgehub/fundamentals/what-is-sparql/.

[17] PRABHU, V. *What is back end development* [online]. YoungWonks, január 2021 [cit. 2022-04-19]. Dostupné z: https://www.youngwonks.com/blog/What-is-Back-End-Development.

[18] RDF WORKING GROUP. *Resource Description Framework (RDF)* [online]. W3, február 2014 [cit. 2022-04-19]. Dostupné z: https://www.w3.org/RDF/.

[19] SKLAR, D. *PHP 7 : praktický průvodce nejrozšířenějším skriptovacím jazykem pro web*. Vydání první. Brno: Zoner press, 2018. ISBN 978-80-7413-363-3.

[20] ULIČNÝ, V. *Proč zkusit vývoj webu a webových aplikací v Reactu?* [online]. Rascasone, júl 2021 [cit. 2022-04-19]. Dostupné z: https://www.rascasone.com/cs/blog/proc-react-vyvoj-web-aplikace.

[21] VOVCHUK, Y. *Srovnání progresivních frameworků VUE.JS, React a Angular*. Praha, CZ, 2020. Bakalářská práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky. Dostupné z: https://vskp.vse.cz/78656.

[22] W3 TEAM. *What is HyperText* [online]. [cit. 2022-04-19]. Dostupné z: https://www.w3.org/WhatIs.html.

Príloha A

Obsah pamäťového média

- **PageView.zip** - obsahuje celú aplikáciu, do ktorej sú implementované moduly z tejto práce
- **xhruzt00.zip** - obsahuje len zdrojové kódy vytvorené pre potreby tejto práce
- **readme.txt** - potrebné informácie k obsahu média
- **xhruzt00.pdf** - finálna verzia tejto práce vo formáte PDF

Príloha B

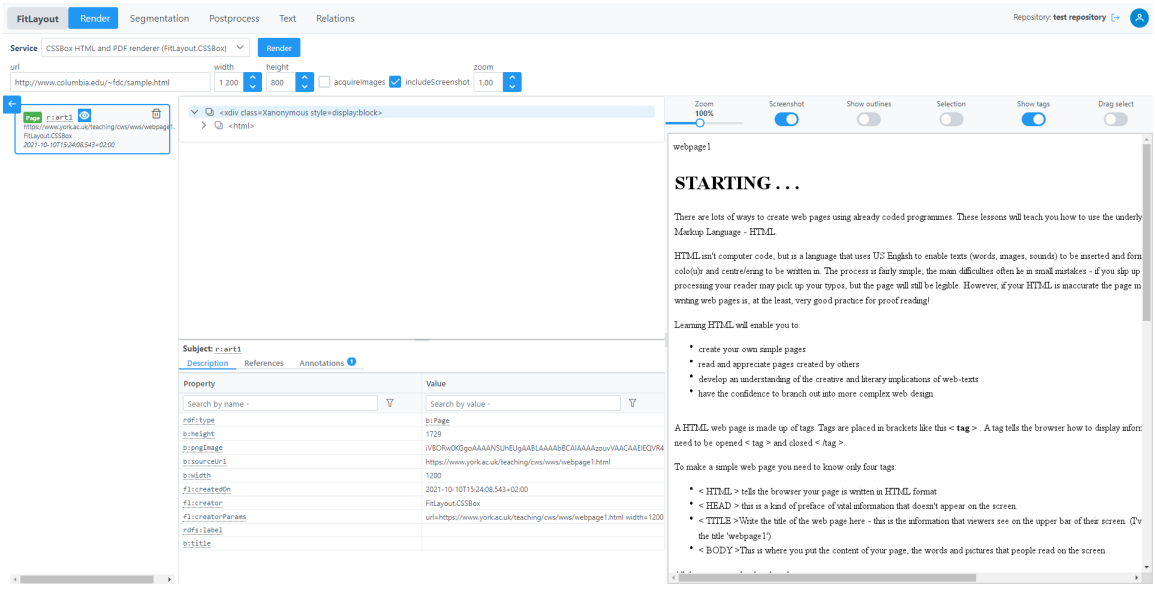
Inštalácia

- Rozšírenie tejto práce sa inštaluje spoločne s celou aplikáciou PageView, ktorá sa nachádza aj s návodom na inštaláciu na: https://github.com/FitLayout/PageView
- Inštalácia je cez docker-images na: https://github.com/FitLayout/docker-images
- Pre správne fungovanie je potrebné spojazdnenie Java backendu tiež v docker-images

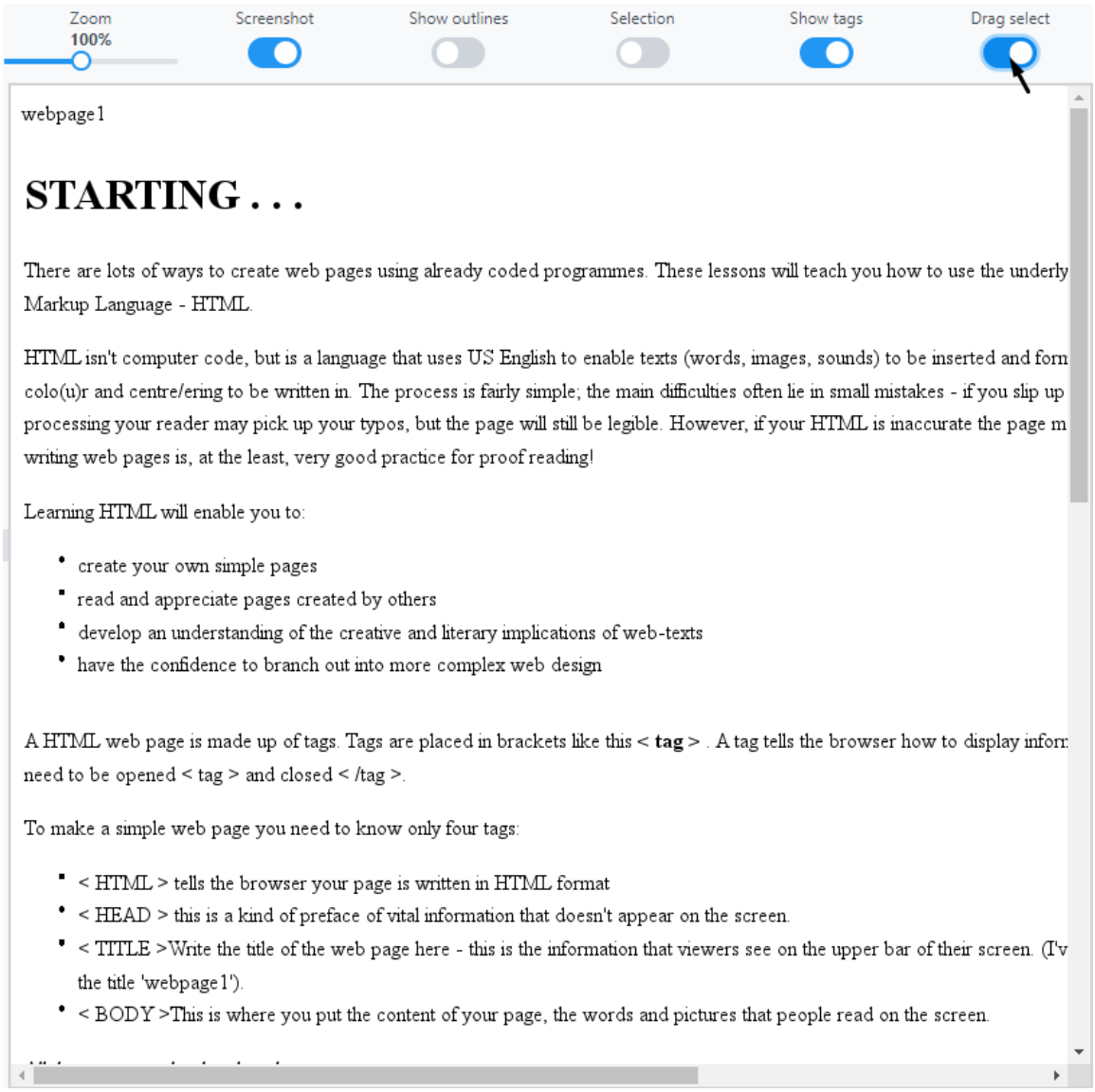
Príloha C

Popis použitia

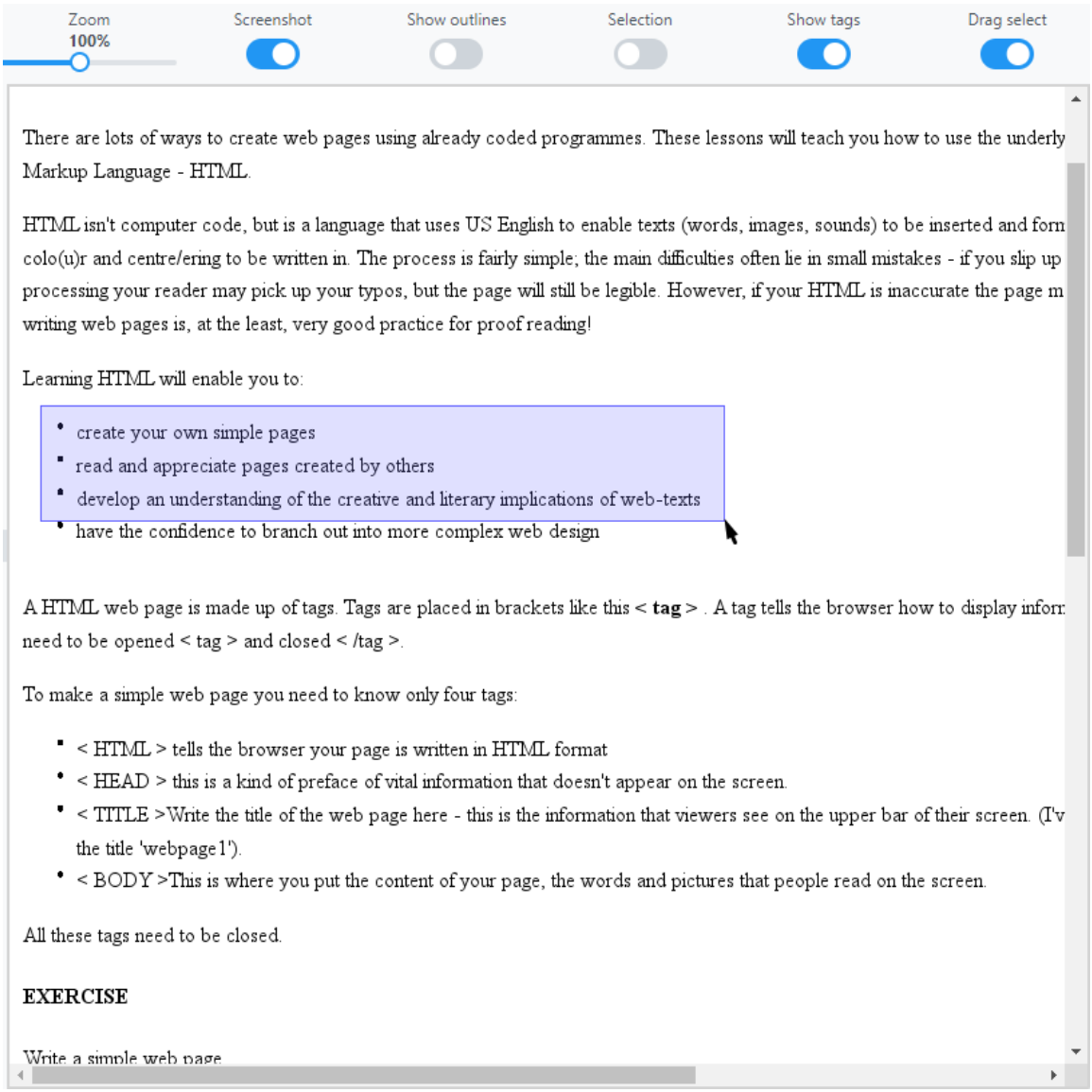
V tejto prílohe je znázornené použitie aplikácie v praxi.



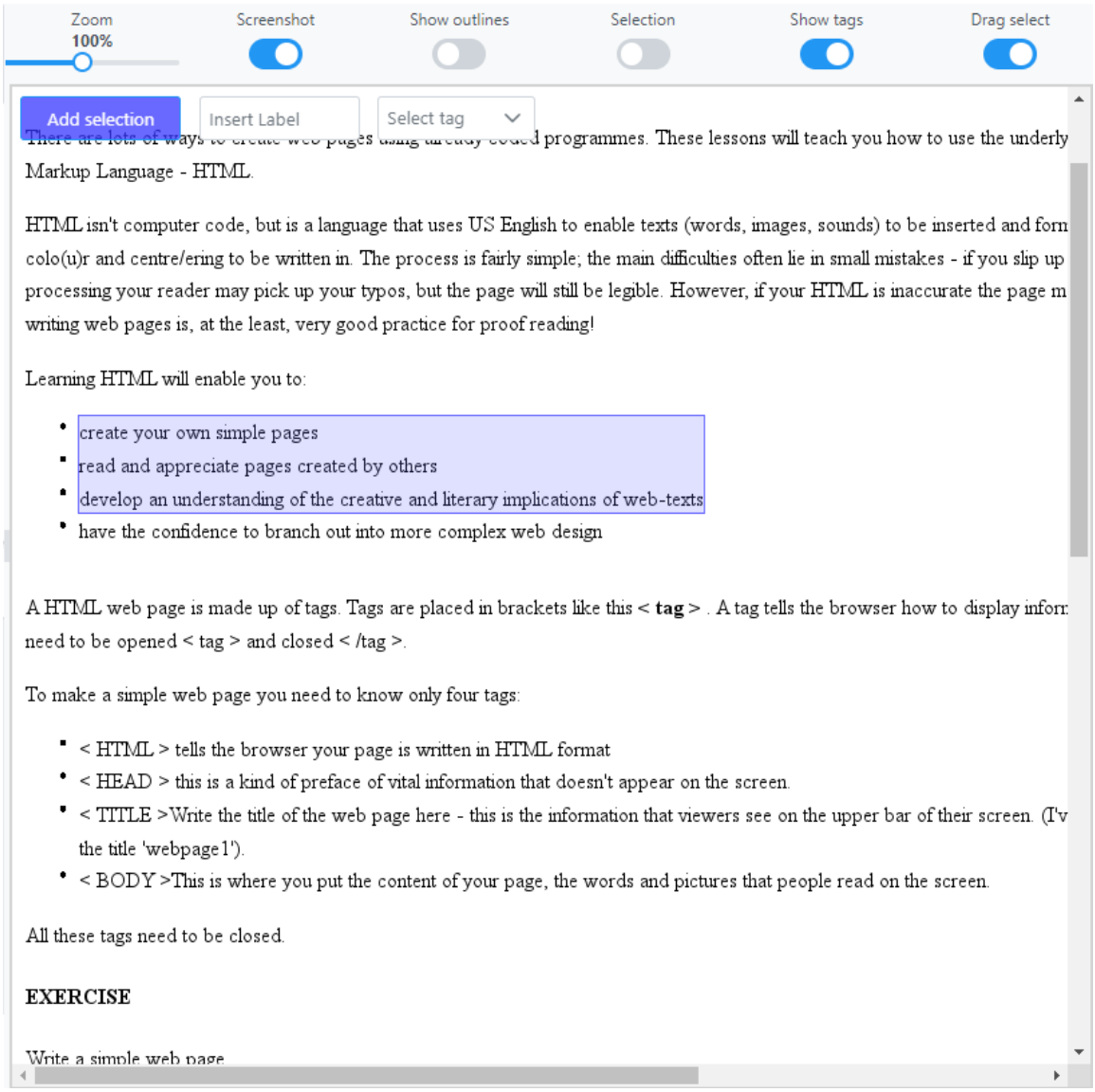
Obr. C.1: Rozhranie s vykresleným webovým dokumentom, stromovou štruktúrov a anotačným menu



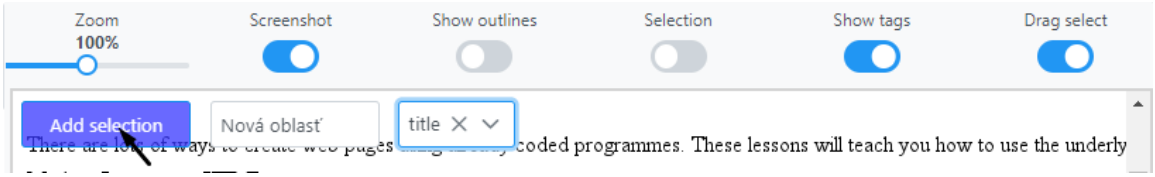
Obr. C.2: Zapnutie možnosti výberu oblastí



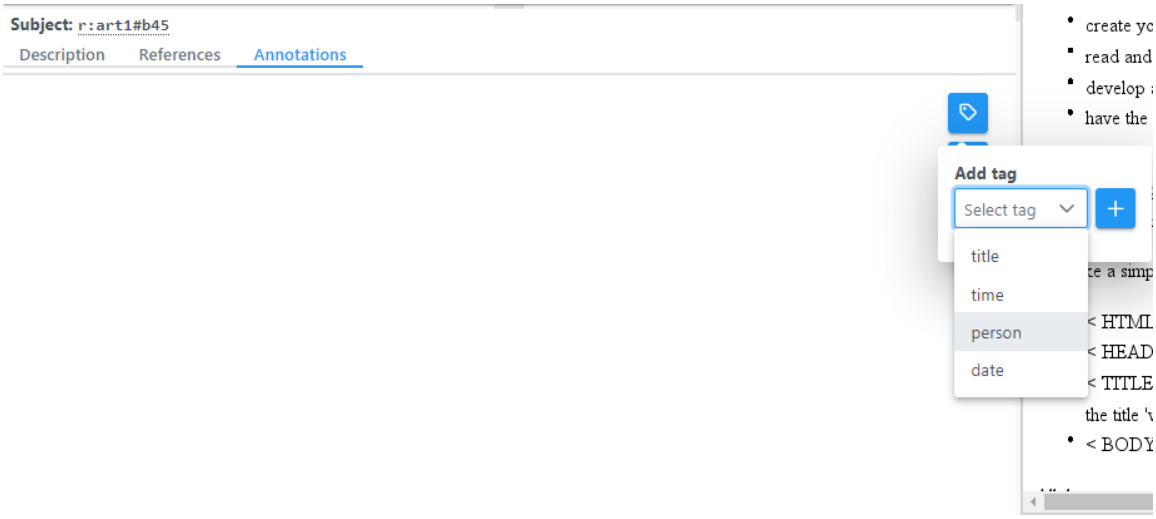
Obr. C.3: Výber oblasti stlačením ľavého tlačidla myši a následným ťahaním



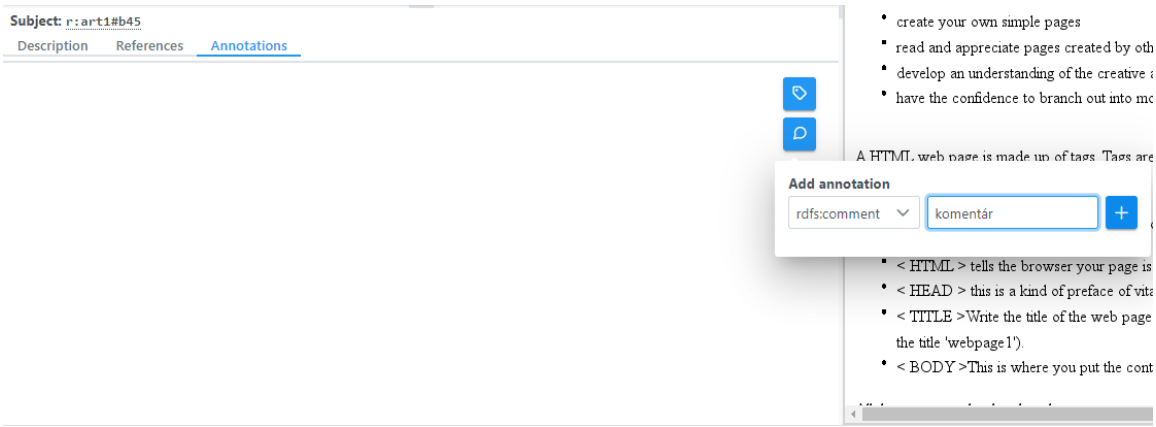
Obr. C.4: Ohraničená vybraná oblasť a zobrazené kontrolné prvky pre pridanie novej oblasti



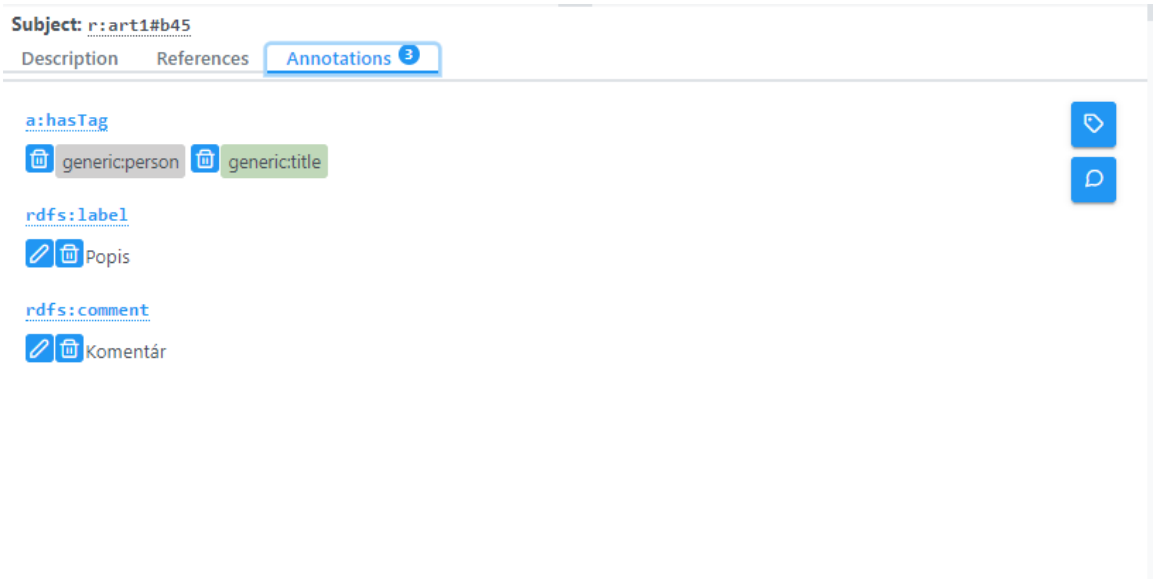
Obr. C.5: Pridanie novéj oblasti po zadaní popisu a značky



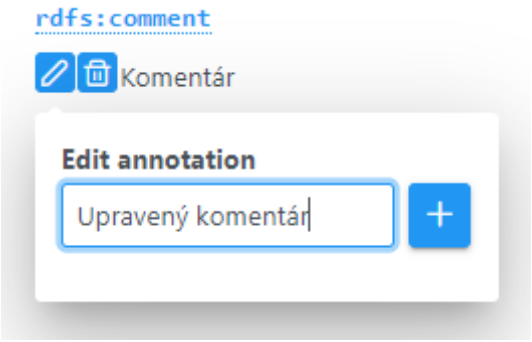
Obr. C.6: Pridávanie nových značiek do artefaktu



Obr. C.7: Pridávanie nových anotácii do artefaktu



Obr. C.8: Artefakt so značkami a pridanými anotáciami



Obr. C.9: Editácia už pridanej anotácie